3A-5

制御機器向け Linux における起動時間短縮手法の一検討

堀井　圭祐†　　攝津　敦†　　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

近年、制御機器はメモリ保護やネットワーク対応などの高機能化が進んでおり、標準的にこれらの機能が搭載されている Linux の適用が拡大している。Linux は多様なサービスをサポートし高機能である反面、制御機器への適用では OS の起動に時間を要するため、高速化が検討されている[1]。

従来は Linux の起動時間を短縮するために H/W 性能やシステム要求に応じて、個別にカスタマイズすることが一般的であったが、複数の制御機器に Linux を適用する場合は、開発量の増大が問題である。そこで、開発量削減のために共通して起動時間を短縮可能な手法について検討を行った。本稿ではその検討結果について述べる。

2.起動時間の測定および分析

2.1. H/W・S/W 仕様

今回、3 種類の PowerPC 基板を用いて起動時間を測定した。表 1、表 2に示すように 3 基板の S/W 仕様は統一しているが、H/W 仕様は異なり基板 1 の性能が高く基板 3 の性能が低い。

2.2. 起動時間の測定

起動高速化する対象は電源 ON から Linux のログインプロンプトが出力されるまでとする。本ターゲットシステムにおける起動シーケンスはブートローダ(U-boot)の起動、kernel の起動、init プロセスの起動に大別できる。そこで、teraterm のタイムスタンプ機能を用いて、コンソールのログメッセージから基板毎に各プロセスの所要時間を測定した。　図 1に起動時間の測定結果を示す。図 1より、全基板に共通して init プロセスに 40%以上

表 1　H/W 仕様

	CPU	メモリ	キャッシュ
基板 1	800MHz	DDR3 32bit 400MHz, 512MB	L1: 32KB L2:256KB
基板 2	400MHz	DDR2 16bit 266MHz, 256MB	L1: 16KB
基板 3	264MHz	SDRAM 16bit 166MHz, 128MB	L1: 16KB

表 2　S/W 仕様

	OS	kernel
全基板	32bit Debian 7.0	3.4.51

An examination of fast boot method for Linux on control devices

Keisuke Horii, Atsushi Settsu, Shinichi Ochiai

† Information Technology R&D Center, Mitsubishi Electric Corporation

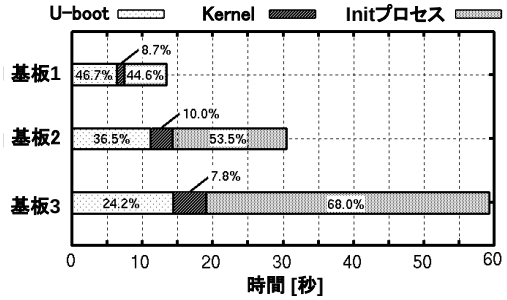

図 1　各基板の起動時間

の時間を要していることを確認した。また表 1と比較して、U-boot の起動時間は CPU の性能差とほぼ同程度の比率であるのに対し、kernel、init プロセスの処理はそれ以上の差が生じていることが判明した。特に init プロセスの時間差が大きく、H/W の性能差に影響され易いと考えられる。U-boot や kernel の処理は H/W に依存しているが、init プロセスでは H/W に依存せず各基板で同様の処理を行っている。そこで H/W 性能差に依存しやすく、処理は同じである init プロセスを改善することで、各種 H/W に適用可能な起動時間短縮が得られると考える。

2.3. init プロセスの分析

init プロセスにおいてボトルネックとなっているスクリプトを抽出するために、bootchartd[2]を用いて起動時間を測定した。測定の結果、sysinit における 3 つのスクリプト (mtab.sh, mountall.sh, networking) の所要時間が長く、基板 1 では init プロセス (図 1における 44.6%)の 50%程度であった。これは他の 2 基板についても同様の傾向であり、基板 2 において 54%程度、基板 3 において 60%程度という結果であった。

2.4. ボトルネックにおける所要時間の要因分析

本ターゲットシステムにおいてボトルネックのスクリプトを抽出したため、所要時間の要因を分析する。はじめに分析結果として、図 2に基板 2 における init プロセスの所要時間の内訳を示す。

(1) mtab.sh

mtab.sh は、/run などの tmpfs および/proc, /sys のマウントや、/etc/mtab と/proc/mounts における整合性を確認するスクリプトである。

mtab.sh 内で最も所要時間が長い処理は、H/W 仕様を確認する grep/sed の実行であり 0.30s (0.03s×10 回)要していた。さらに、mtab.sh は内部で別のスクリプトを 4 回呼び出しており、呼び出されたスクリプト内でも grep/sed 処理が実行されているため mtab.sh の 32%にあたる約 1.50s 要していた。

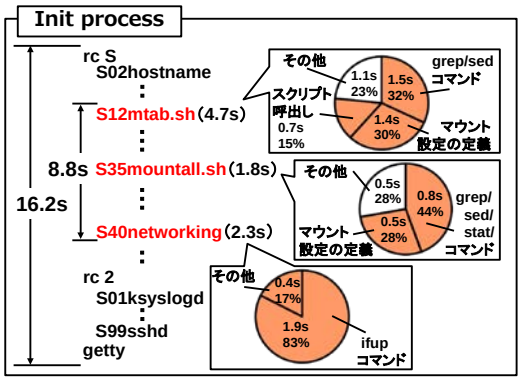

図 2　init プロセスの所要時間の内訳

mtab.sh はマウントの前処理として複数の制御文を用いて、オプションなどを定義している。例えば、/etc/fstab から設定を読み込む際には while 文を用いて走査している。これらマウントの設定を定義する処理の時間は平均 0.07s であり合計 19 回実行されているため、mtab.sh の 30%にあたる約 1.40s 要していた。

mtab.sh は前述したように別のスクリプトを 4 回呼び出しており、1 度の処理時間は 0.18s であった。これにより mtab.sh の 15%にあたる約 0.72s 要していた。

(2) mountall.sh

mountall.sh は、ローカルファイルシステムのマウントなどを行うスクリプトである。

mountall.sh における所要時間の主な要因は、図 2に示すように mtab.sh と概ね同様であった。高機能なコマンドの所要時間については、grep/sed 処理の実行時間 0.30s に加えて、処理時間が 0.06s である stat コマンドが 8 回実行されているため、mountall.sh の 44%にあたる約 0.8s 要していた。また、マウントの条件を定義するための処理は 8 回実行されているため、mountall.sh の 28%にあたる約 0.50s 要していた。

(3) networking

networking では interface を立ち上げるために、networking の 83%にあたる 1.9s を ifup コマンドの実行に要していた。

3. 制御機器向けの起動時間短縮手法とその改善効果の見込み

3.1. スクリプト内容の固定化

図 2における mtab.sh, mountall.sh での高機能コマンド (grep/sed など) は H/W・S/W 仕様の変化に追随するために実行されているが、制御機器の場合には仕様が動的に変化することはないため、これらの処理は省略可能である。同様にマウントするための設定を定義するための処理についても、マウントするファイルシステムの設定は固定であり省略可能である。また、mtab.sh は複数の異なるスクリプトを呼び出すために時間を要しているが、1 つのスクリプトにまとめることで時間を削減可能である。

本稿では複数の基板に共通した起動時間短縮を目的としている。そこで H/W・S/W 仕様などはスクリプト内で定義するのではなく、各基板向けに設定ファイルを用意し、設定ファイルのみ変更することで各基板に適用させる。

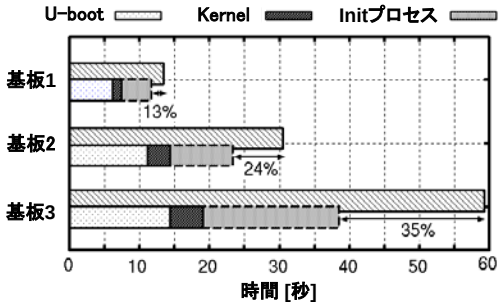

図 3　起動時間短縮後の見込み値

3.2. 起動時の実行スクリプトを削減

図 2より networking は、ifup コマンドの実行に時間を要しているが、ifup コマンドは interface の立上げに必須のコマンドであり削減することはできない。しかし、ネットワークの利用は起動直後ではなくアプリケーション実行時に必要とされるため、アプリケーションで利用する直前に立ち上げれば良い。そこで、networking を起動時のスクリプトから除外することで所要時間を削減可能である。このように、起動時間に大きく影響のある処理については、要否を判断し起動後に実行するよう改善することで起動時間を削減可能である。

3.3. 改善効果の見込み

図 2に示す円グラフ中の塗りつぶされた部分は、3.1, 3.2節で述べたように制御機器においては削減可能な処理であり、基板 2 において起動時間の 24%程度を占める。つまり、基板毎の設定ファイルに基づいてスクリプトの内容を固定化、起動時の不要な処理を削減することにより 24%程度の起動時間短縮が見込まれる。各基板において、ボトルネックとなるスクリプトは共通しており、削減可能な処理の割合を測定したところ基板 1 において 13%、基板 3 において 35%であった。各基板における起動時間短縮後の見込み値を図 3に示す。図 3より init プロセスを改善することによる起動時間短縮の効果は、H/W 性能が低いほど大きく実装前と比較して基板間の時間差が減少している。

4. おわりに

本稿では、制御機器向けの組込み Linux における起動時間短縮のために、3 種類の PowerPC 基板を用いて起動時間を測定・分析した。分析の結果、全基板に共通して init プロセスの所要時間が長く、全体の 40%以上であった。そこでボトルネックとなるスクリプトを抽出し、複数の制御機器に対して基板毎の設定ファイルに基づいて不要な処理を削減する手法を検討した。本手法を用いることによって 15～35%程度の改善が見込まれる。

今後は各基板に本手法を実装し評価することで、基板毎の差異を分析する予定である。

参考文献

[1] 出原、攝津、伊藤、落合：「コンシューマ向け組込み機器のアプリケーション起動高速化検討」、情報処理学会 第 67 回全国大会(2005)

[2] Bootchartd : http://elinux.org/Bootchart