Vol.2011-OS-119 No.11
Vol.2011-EMB-23 No.11
2011/11/29

組込みシステムを指向した Linux プロセスログ取得機構

Praween Amontamavut[†]　中川 裕貴[††]　早川 栄一[†]

組込みシステムを対象とした省メモリの Linux プロセスのロギング機構の開発を行った．組込みシステムでシステム動作のロギングを行う場合，メモリ容量に限界があるため，リアルタイム処理を含むログデータを取得する際にコンテキストスイッチのログの消失が起こる可能性がある．この問題を解決するために低オーバヘッドで，小メモリ容量の環境下でログを生成，取得が可能な圧縮機能付きトレース機構を提案した．具体的には，Linux が提供している ftrace に対してコンテキストスイッチログを圧縮する機能を追加した．また，ユーザ空間へのエクスポートを行わずに外部へ転送することで， コンテキストスイッチへの影響を減少させた．

Linux Process Logging Mechanism for Embedded Systems

Praween Amontamavut[†]　Yuki Nakagawa[††]
and Eiichi Hayakawa[†]

We have developed a process logging mechanism for Linux OS that utilizes a small memory embedded systems environment. Taking context switch log from operating system, A part of log data including real-time processing log data may be lost by limited memory. Solving this problem, we propose a low overhead tracing mechanism with compressing trace log data in a small memory environment. The compression mechanism is added to Linux ftrace mechanism. The influence to switching processes is decreased by transferring the traced data to external server without exporting to user space.

1. はじめに

現在, Linux カーネルをベースとした Android は組込みシステムで多く用いられている．Android を組込みシステムに用いる場合，スケジューリングや OS のデバッグは問題となっている．この原因の一つに，Android ではシステムが OS と DalvikVM からなる多層構造であり，スケジューラや VM の動作がどのように影響するのかがわかりにくいという点がある．[(1)]

このことから，アプリケーションの実行速度やヒューマンインタフェースの応答速度の向上には，Android 上で動作するプロセスのスケジューリングについて工夫が必要となる [(2)(3)(4)] Android 上で動作するプロセスは，Linux のプロセスに関連づけられることから，Linux カーネルのスケジューリングの学習や検証，プロセススケジューリングのデバッグも重要になっている．

しかし，組込みシステムのデバッグでは, 動作の再現が難しいことや，システム環境自体のよる影響が大きい．[(5)] [(6)]このことから，アプリケーションの開発や学習，検証においては，実行の動作をロギングし，それを分析や可視化することが有効な手段となる．

拓殖大学早川研究室では, カーネルプロセス及びユーザプロセスを学習するために, 2009 年から Linux および Android を対象としたプロセスの可視化について研究を行ってきた．[(7)]プロセススケジューリングのデバッグは, Linux に内蔵されているトレーサを用いて, コンテキストスイッチのログを取得する．プロセスの可視化も，コンテキストスイッチのログを取得し, 保存してから解析を行う．システムの実時間性を監視するには，長時間のコンテキストスイッチログが必要になる．

しかし, 生成されたコンテキストスイッチのログのサイズは, 短時間であっても大きいことから, バッファの容量が少ない組込みシステムでは, 問題が発生する．その理由としては, コンテキストスイッチのログを生成する速度が, 取得する速度よりも速いことにある． Linux のプロセスロギング機構では, リングバッファを用いて速度差を吸収するが, データのサイズが大きい場合, 過去のデータが上書きされてしまい, ログ情報の一部の情報が消失する可能性がある．

また, データの外部への転送では, それ自体にオーバヘッドがあり, これもプロセスのコンテキストスイッチに影響を与える可能性がある．

本研究の目的は, ログを生成する処理から表示するまでにオーバヘッドが少ない, プロセスログ取得方法や機構改良方法を提案し,実装, 評価することである．

[†] 拓殖大学　工学部　情報工学科
Faculty of Engineering, Department of Computer Science, Takushoku University
[††] 拓殖大学　大学院　電子情報工学専攻
Graduate School of Engineering, Electronics and Information Science, Takushoku University

Vol.2011-OS-119 No.11
Vol.2011-EMB-23 No.11
2011/11/29

2. Linux の ftrace 機構

2.1 ftrace 機構

Linux が提供しているトレース機能である ftrace の構造を図 1 に示す. トレース機構は，トレース内部機構とトレース外部機構の二つに分割されている.

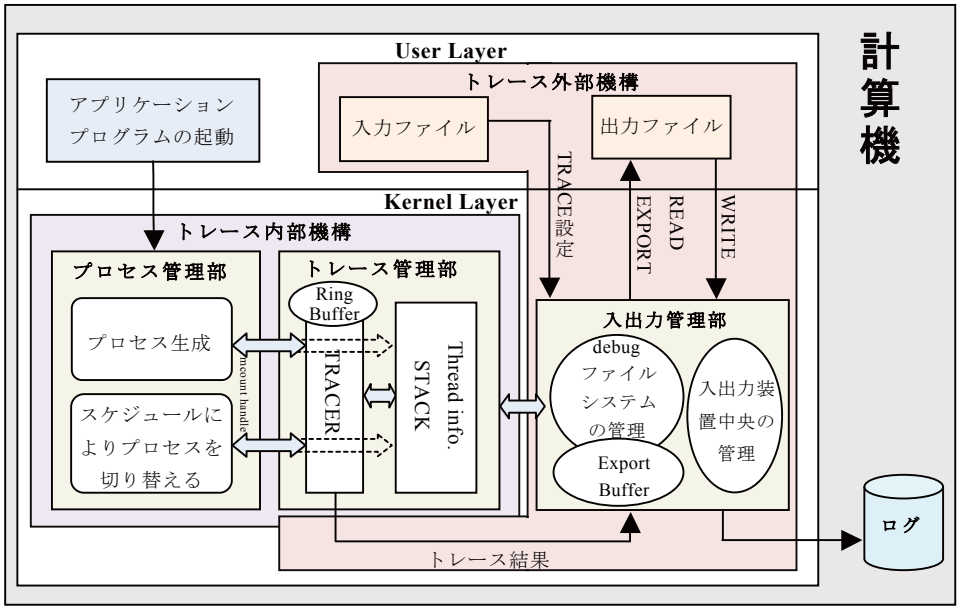

図 1 従来のトレース機構

トレース内部機構は，プロセススケジューリングを管理するプロセス管理部からコンテキストスイッチに関するログ情報をバイナリデータのまま取得し，リングバッファにノンブロッキングで格納する.

トレース外部機構は，ユーザ空間とカーネル空間の両方に存在する．トレース内部機構からユーザ空間にあるプロセスに，ログデータをエクスポートさせる機能を用いる．これをログデータエクスポート部と呼ぶ．ログデータをエクスポートする機能は，debug ファイルシステムを介して前者にログデータ要求を行う．要求に対して，OS はリングバッファからエクスポートバッファにブロッキング方式でログデータを収集し debug ファイルシステムを通じてログデータをユーザ空間にエクスポートする.

2.2 コンテキストスイッチのログ

コンテキストスイッチのログは，カーネルによるプロセススケジューリングの追跡を目的としたシステムのログであり，プロセス生成やプロセス状態の切換えなどの情報を保持しているシステムのログである.

このログは 2.1 で述べたように，リングバッファに格納されるバイナリデータを，容易にログデータを分析できるように，図 2 のようにデータを文字列化し，エクスポートバッファに収集し，ユーザ空間にエクスポートする形になっている.

表 1 に，空白で区切った一行のログの要素でコンテキストスイッチのログを分析した例を示す．プロセス名，発生した時間と，スイッチ前後のプロセスの状態を取得することができる.

```
sh-42    [000] 4154504421.591616:    42:120:S    + [000]    42:120:S
sh-42    [000] 4154504421.591616:    42:120:S    ==> [000]    0:140:R
```

図 2 コンテキストスイッチのログの例

表 1　コンテキストスイッチのログの意味

Group Element	各要素の意味
PROCESS DESCRIPTION	{プロセス名}-{プロセス ID}
PRESENT CPU	[現在の CPU]
ENTRY TIME	{エントリ時刻秒単位}.{エントリ時刻マイクロ秒単位} :
PRESENT PROCESS DETAIL	{現在のプロセス ID} : {現在の優先度} : {現在のプロセス状態}
PROCESS ENTRY SIGN	{エントリの種類}
NEXT CPU	[次の CPU]
NEXT PROCESS DETAIL	{次のプロセス ID} : {次の優先度} : {次のプロセス状態}

2.3 従来の機構の問題点

従来のトレース機構の問題点は次の 2 点である.

(1) ユーザ空間へデータを転送するために OS が用意しているエクスポートバッファのサイズは，1 ページである．コンテキストスイッチのログは，冗長でサイズが大きい文字列に変換してエクスポートすることから，生成と取得との速度差

Vol.2011-OS-119 No.11
Vol.2011-EMB-23 No.11
2011/11/29

を吸収できない可能性がある．その結果，リングバッファからエクスポートバッファへのデータコピーがリングバッファのデータ収集速度に間に合わずに，プロセスのログデータが消失するという問題が発生する．

(2) ログデータのユーザ空間へのエクスポートは，debug ファイルシステムを一度通過する形となっているので，ファイルシステムのオーバヘッドがある．また，組込みシステムは機器内の Flash ROM の容量に制限があることから，機器内に大容量のログデータを保存することはできない．

3. 本研究のトレース

3.1 トレース機構の設計

本機構の構成を図３に示す．本機構は，ログ生成側の組込み機器とログ取得側のログ収集計算機の二つから構成する．ログを生成する計算機は，従来トレースと同様にログを生成しながら，データを外部にエクスポートする． ログを生成する側では，データ圧縮することで，保存できるログの容量を増やす．また，収集したログは，ユーザ空間へコピーすることなく，カーネルから直接ログ収集計算機にコピーする．これによって，プロセス切替えへの影響を減らすことができる．

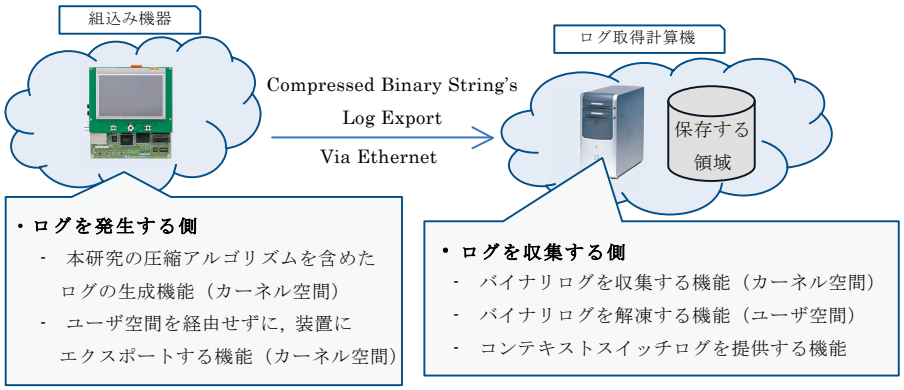

図 3 トレース機構の構成

図 4 にトレース外部機構を中心として機構を改良した本研究のシステムの機構を示す．本システムでは，従来のトレースと同様にトレース外部機構とトレース内部機構に分割する.プロセス管理部との関連性が高いことから,トレース内部機構で用いている構造体を再設計することは, Linux の移植性やバージョンアップが困難になる．そこで，トレース外部機構を中心に改良を行う.トレース外部機構に圧縮部を追加する．さらに，直接データを外部装置にエクスポートできるように，トレースドライバを追加する．

ログを収集する計算機では，組込み機器からのログを取得し保存する．そして，ログ解凍機能により圧縮されたログを解凍し，利用者がアクセスできるようにする．

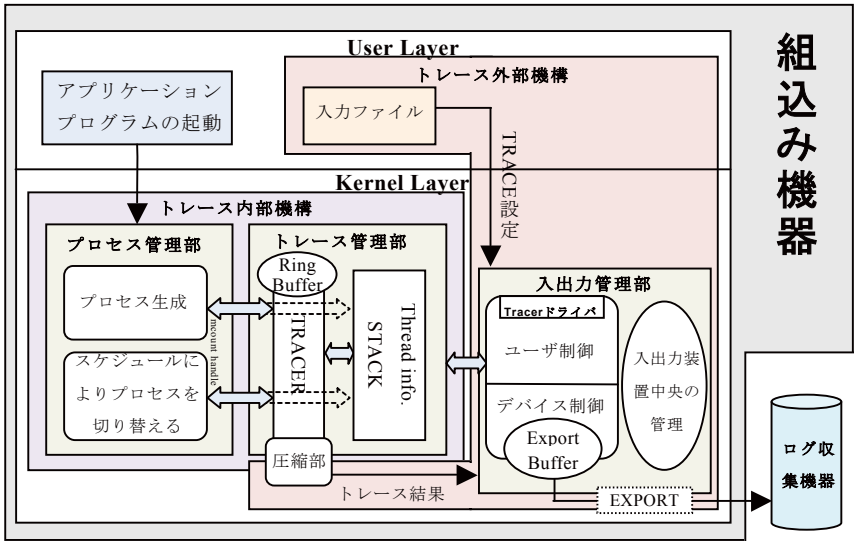

図 4 ログ取得側のトレース機構

3.2 圧縮機能の設計

圧縮機能は，リングバッファから出力されたデータを，小さい容量のエクスポートバッファで扱えるようにする．圧縮の手法は，従来のトレースの出力ログデータの特徴を分析し，データフォーマットの特徴を活かす方法を用いる．コンテキストスイッチのログの特徴は三つである．

(1) データ表現の範囲が広く，同じデータの繰り返し頻度が高いデータ
(2) データ表現の範囲が狭いデータ
(3) データ表現の範囲が元のデータサイズに最適となっているデータ

三つの特徴によって，エクスポートバッファに蓄える一行のログを図 5 のように文字列部とバイナリ部に分割する．

Vol.2011-OS-119 No.11
Vol.2011-EMB-23 No.11
2011/11/29

文字列部は不要な文字を削除した文字列データのグループとする．バイナリ部は，データサイズが最も小さい固定サイズのバイナリ部をヘッダバイナリデータにし,メモリ上に占有する時間を減らすため後方に配置する．

そして，解析したログの特徴に従ってデータの圧縮を行う．(1) の特徴のデータは，そのデータの文字列の長さを示す適切な範囲で表現しているバイナリデータを持っているデータを，文字列可変長データにする．文字列可変長の解析は次に述べる．

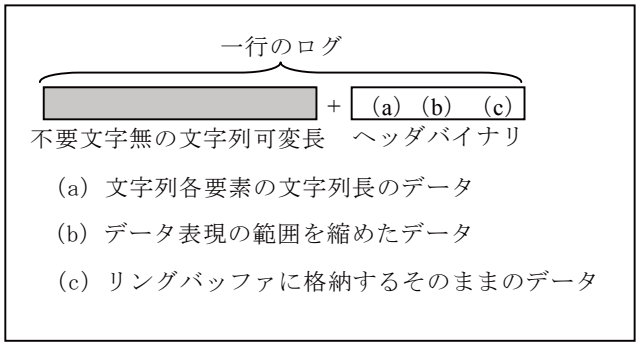

図 5 圧縮機能によりエクスポートバッファに格納する一行のログ

二回以上連続するデータであれば，その二回目以降のデータを前者に書き込まないようにし，長さを示す (a) に 0 を書き込む．その他はそのままに文字列部に書き込み，その長さを (a) に書き込む．(2)の特徴のデータは表現する範囲を縮め，バイナリ部にバイナリコードを書き込む．(3)の特徴のデータは何もせずにそのままバイナリ部にバイナリコードを書き込む．

3.3 解凍方法

データの末尾から固定サイズのヘッダバイナリを読込み，圧縮機能によりエクスポートバッファに格納する一行のログを表す図 5 による (a) ～ (c) の各データを一単位ずつ，図 6 に示す中間ログ形式に変換し，一行ずつに解析する．

中間ログは最終ログに近いデータ形式であるが，データが存在しない部分も残した状態のログである．

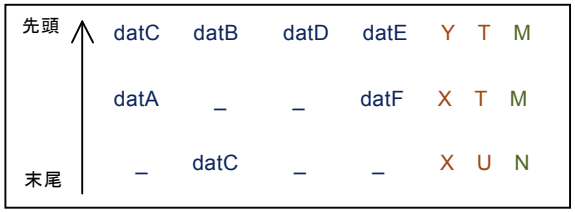

図 6 中間ログのイメージ

(a) のデータ部分は各要素の文字列の長さをバイナリ部から読み込み，各要素の文字列を分離する．文字列の長さが 0 の場合は，文字列部にデータがエクスポートされていないことから，要素分離困難を回避するため“_”に格納する．そうではない場合は，その要素の長さだけの文字列の範囲で格納する．(b)と(c)のデータ部分はそのまま文字列化し，格納する．

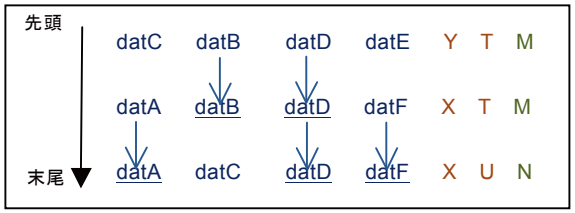

図 7 解凍した最終ログのイメージ

図 7 は文字列コードで表現するコンテキストスイッチ形式や JSON の形式の最終ログを表す．解析された中間ログを図 7 のように先頭から読み込んで，データ追加処理を行い，システム標準のコンテキストスイッチログの形式や JSON の形式の最終ログに解析し生成する．

Vol.2011-OS-119 No.11
Vol.2011-EMB-23 No.11
2011/11/29

4. 圧縮付きトレースの評価

4.1 評価システムの環境

本研究はKZM-A9評価ボード上に動作するカーネルバージョン2.6.29をベースにシステムプロトタイプを実装し評価する．環境の詳細は表2に示す．ARM Cortex-A9では，1ページは4096バイトである．

表 2　評価環境

Hardware	KZM-A9 評価ボード	
	CPU	ARM Cortex-A9MPCore™ Dual CPU
		Operating Frequency 533MHz (MAX)
		CPU Data cache 32KB/CPU
		CPU L2 cache 256KB
	Memory	Main Memory 512MB(DDR2-533)
		Internal SRAM 128KB
		Internal ROM 64KB
		eMMC NAND 4GB
Software	Linux Kernel2.6.29 上に動作する Android2.2	

4.2 評価解析

評価する点としては，ログデータ収集の精度を表すデータ収集効果と，データを生成されたデータのサイズ当たりのシステム実行時間を表すデータ生成オーバヘッドの２点である．

データ収集効果は，圧縮機能の有無によって，ログデータを収集する効果を比較する．これを式(A)で表す．

$$\text{ログデータ収集効果} = \frac{\text{使用ログデータサイズ}}{\text{プログラム実行時間}} \quad \cdots\cdots (A)$$

データ生成オーバヘッドはデータ収集効果に対するオーバヘッドを表す．圧縮機能を追加するトレースによるログを生成し収集するまでのオーバヘッドを，圧縮機能を追加しない従来のトレースの場合のそのオーバヘッドと比較する．データ生成オーバヘッドは式(B)に表す．

$$\text{データ生成オーバヘッド} = \frac{\text{プログラム実行時間}}{\text{生成ログデータサイズ}} \quad \cdots\cdots (B)$$

生成ログデータサイズはリングバッファに格納し，エクスポートされるログデータのサイズであり，使用ログデータサイズはエクスポートされたデータを実際に使用するログデータのサイズである．

4.3 評価

今回は従来のトレースと本研究で実装した圧縮付きトレースを評価する．表2に示す環境で，圧縮機能付きおよび圧縮機能なしのシステムに対して，Android をランダムな時間実行し，生成ログデータサイズおよび使用ログデータサイズを取得した．そのデータを元に式(A)と式(B)に従って計算し,グラフにプロットした結果を 図8と図9に示す．

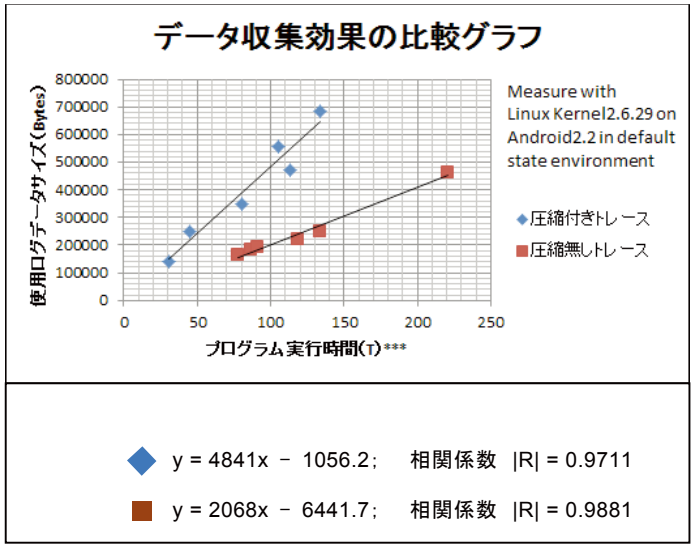

図 8 データ収集効果の比較グラフ

Vol.2011-OS-119 No.11
Vol.2011-EMB-23 No.11
2011/11/29

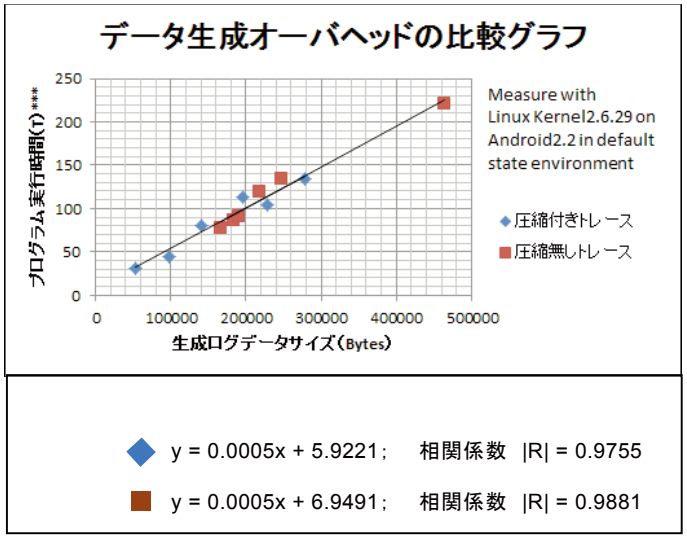

図 9 データ生成オーバヘッドの比較グラフ

グラフから，圧縮機能付きトレースは圧縮機能なしの従来トレースよりも 2 倍ほどデータ収集効果が高く，オーバヘッドも従来トレースとは変わらないことが明らかになった．このことから，圧縮付きトレースはサイズの小さいエクスポートバッファに対して圧縮機能により圧縮することで，エクスポートバッファに保存できるデータサイズが増加し，リングバッファにより多くのデータを収集できることが分かる．

この結果から，圧縮機能付きのトレースはバッファが小さい環境の下であっても，データ収集効果を上げ，データ消失を防ぐことができることが明らかになった．

5. おわりに

本報告では，Linux の ftrace 機構に圧縮機構を追加し，ユーザ空間へコピーすることなく，ログデータを転送するトレース機構について述べた．この機構により，組込みシステムのような小さな記憶容量しか持たないシステムであっても，ftrace 機構を用いて，プロセスのロギングが可能になった．実際に Android 上で本システムを評価し，少ないオーバヘッドで，従来の ftrace で保存できるログの 2 倍程度のログの保存が可能になった．これによって，圧縮機能付きのトレースはバッファが小さい環境の下であっても，データ収集効果を上げ，データ消失を防ぐことができることが明らかになった．

今後の課題は，他の圧縮方法との比較，およびプロセス以外のロギングデータについての扱い，および外部へのデータ転送に関する評価である．

参考文献

1) S. Vijay Anand, Kumar S.P Suman, “A heuristic buffer management technique on android to enhance video quality on digital handheld audio video terminals”, The IEEE symposium on Computers and Communications, 2010
2) Matthew S. Jaffe, Nancy G. Leveson, Mats P.E. Heimdahl, and Bonnie E. Melhart , “Software Requirements Analysis for Real-Time Process-Control System”, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL 17, NO. 3, MARCH 1991, http://users.ece.gatech.edu/~bonnie/courses/ComputingControl/software_requirements_control.pdf
3) RTSS2011, “Design and Verification of Embedded Real-Time Systems” , http://cse.unl.edu/rtss2008/index.php?SelectedItem=DesignAndVerification_CFP&SelectedGroup=CallForPapers
4) Steven Cavanagh, Yingxu Wang, “Design of a Real-Time Virtual Machine (RTVM)”, http://coe.uncc.edu/~jmconrad/ECGR6185-2008-01/notes/Real%20Time%20Virtual%20Machine.pdf
5) Midori Sugaya, “Online Log Analysis System for Real-time System’s Faults”, Work-in-Progress Session, RTCSA 2011
6) M. of Economics., “Annual report in 2007”, embedded system software industry report(in Japanese)
7) 中川裕貴，早川栄一，西野洋介, “Android における OS プロセス可視化環境の開発”, 情報処理学会組込みシステム研究会報告，2011-EMB-21(3),1-6, 2011