社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－SLDM－139 (12)
2009－ＥＭＢ－ 12 (12)
2009／3／6

入出力機器の動作履歴を考慮したドライバプログラム起動制御法の評価

滝口 真一† 谷口 秀夫††

† †† 岡山大学大学院自然科学研究科 〒 700–8530 岡山県岡山市津島中三丁目 1 番 1 号
E-mail: †takiguchi@swlab.cs.okayama-u.ac.jp, ††tani@cs.okayama-u.ac.jp

あらまし 入出力機器の種類の増加に伴い，ドライバプログラムの種類数は増加している．このため，ドライバプログラムが占有するメモリ量は増加し，計算機の立ち上がり時間は長大化する．そこで，我々は，入出力機器の動作履歴を考慮したドライバプログラムを起動する制御法を提案した．本稿では，制御法の評価結果を報告する．具体的には，種々の利用パターンに対する制御の有効性を明らかにする．
キーワード ドライバ，マイクロカーネル，適応制御

Evaluation of a Method for Control of Driver Program Invocation based on Device Operation History

Shinichi TAKIGUCHI† and Hideo TANIGUCHI††

† †† Graduate School of Natural Science and Technology, Okayama University
1–1, Tsushima-Naka, 3–Chome, Okayama, Okayama 700–8530 Japan
E-mail: †takiguchi@swlab.cs.okayama-u.ac.jp, ††tani@cs.okayama-u.ac.jp

Abstract Recently, operating systems have become increasingly multi-functional by the providing of various devices. Its functions are provided as driver programs. Even though average users do not use all functions, they do not customize and shrink their operating systems because of its job difficulty. Accordingly, they usually use wasteful and fat operating systems: much memory, slow down bootup time. Consequently, we proposed an effective method for controlling invocation timing of device driver programs. As a strategy for the control, we use operation history on device drivers. This paper describes an evaluation of this method.
Key words driver, microkernel, adaptability

1. はじめに

入出力機器の種類の増加に伴い，ドライバプログラム (ドライバ) の種類数が増加している．モノリシックカーネル構造の OS では，OS 内にドライバを組み込む．このため，利用環境に合わせた OS を構成するためには，適切なドライバの組み込みが必要になる．しかし，ドライバの組み込みには OS の再構築を必要とし，専門的な知識を必要とする．また，工数も大きい．このため，多くの場合，標準的な構成で構築された OS を利用する．したがって，利用者の環境においては，不要なドライバを保有する OS を利用することになり，OS の占有するメモリ量は大きく，計算機の立ち上がり時間は長い．

これらの問題に対処するため，モノリシックカーネル構造の OS では，ローダブルカーネルモジュール (LKM) を用いている．これにより，LKM として作成されたプログラムは，走行中に OS へドライバの組み込みおよび取り外しを行える．また，マイクロカーネル構造の OS においても，ドライバをプロセスとして動作させれば，走行中にドライバの組み込みおよび取り外しを行える．したがって，利用環境に合わせてドライバを組み込むことが可能である．しかし，既存 OS では，長時間利用されなくても，また，入出力機器を取り外しても，ドライバの取り外しは行われない．つまり，利用者による入出力機器の使い方 (利用パターン) を考慮していない．このため，利用されないドライバにより，OS が占有するメモリ量は増加し，計算機の立ち上がり時間は長くなってしまう．

そこで，入出力機器の動作履歴を考慮し，ドライバを起動する制御法を提案した [1]．本稿では，シミュレーションにより，種々の利用パターンに対する制御の有効性を明らかにする．また，制御法を ***AnT*** オペレーティングシステム [2] に実現し評価する．さらに，ドライバの実現形態 (OS 一体型，LKM，プロセス) と占有メモリ量の関係について述べる．

2. 入出力機器の動作履歴を考慮したドライバプログラム起動制御法

2.1 目的と課題

入出力機器の動作履歴を考慮したドライバプログラム起動制御法 [1] の目的を以下に示す.

(目的 1)　ドライバが占有するメモリ量を削減する

(目的 2)　利用開始の要求から利用可能になるまでの時間を短くする

ドライバが占有するメモリ量を削減する制御により，占有メモリ量の削減と立ち上がり時間の短縮を期待できる．占有メモリ量を削減する方法として，ドライバを利用しようとした時点でドライバを起動する制御が考えられる．しかし，これは，ドライバの読み込みと初期化処理により，利用可能になるまでの待ち時間が発生する．このため，利用開始の要求から利用可能になるまでの時間を短くすることを (目的 2) として加える．

上記の目的を達成するためには以下の課題がある．

（課題 1）　多様な制御内容の導入

（課題 2）　制御内容の選択方法

2.2 対　　処

2.2.1 多様な制御内容の導入

必要なときに必要なドライバのみ動作させるため，ドライバの読み込み，初期化，および終了の処理に着目し，制御する．5 つの制御内容の開始契機を図 1 に示し，以下に説明する．

(1)　事前読込：デバイス検出前にドライバを読み込み，デバイス検出時にドライバを初期化する．

(2)　即時起動：デバイス検出時にドライバを読み込み，初期化する．

(3)　遅延起動：デバイス検出時にドライバを読み込まず，その後の空き時間に読み込みと初期化を行う．

(4)　利用要求時起動：AP から利用開始要求が来た時にドライバを読み込み，初期化する．

(5)　自動削除：利用終了要求後にドライバを終了させる．

2.2.2 制御内容の選択方法

適切な制御内容を選択するため，ドライバの動作履歴を収集する．収集する情報を以下に示す．

(1)　履歴収集期間 (H)：動作履歴を収集している期間

(2)　利用開始時間 (S)：デバイス検出されてからの最初の利用開始要求が来るまでの時間

(3)　利用度 (U)：Ut 時間内の利用開始の回数

(4)　非利用時間 (N)：最後に利用されてから現時刻までの時間

制御内容を選択するため，閾値を設ける．なお，利用開始時間は，計算機の起動ごとに一度だけ収集する．このため，利用開始時間については，制御内容選択のため比較を行うとき，前回の計算機起動時の情報を用いる．制御内容の選択方法を表 1 に示す．

3. 評価の観点

3.1 評 価 尺 度

制御法を以下の 3 つの尺度で評価する．

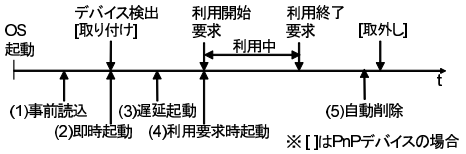


図 1　制御内容の開始契機

表 1　制御内容の選択方法

制御内容	説明	条件	
即時起動	動作履歴の情報が十分でない	$H \leqq H_{th}$	無し
	デバイス検出後短期間で利用し，利用度は低い	$H > H_{th}$	$S \leqq S_{th}$ $U \leqq U_{th}$
事前読込	デバイス検出後短期間で利用し，利用度は高い		$S \leqq S_{th}$ $U > U_{th}$
利用要求時起動	デバイス検出後，利用まで間があり，利用度は低い		$S > S_{th}$ $U \leqq U_{th}$
遅延起動	デバイス検出後，利用まで間があり，利用度は高い		$S > S_{th}$ $U > U_{th}$
自動削除	最後に利用されてから間があり，利用度は低い		$N > N_{th}$ $U \leqq U_{th}$

(1)　占有メモリ量

(2)　立ち上がり時間

(3)　利用開始の即時性

ここで，立ち上がり時間は，ドライバの起動処理開始から，立ち上がり時に起動されるドライバの読み込み，および初期化の処理時間がすべて完了するまでの時間である．また，(目的 2) の満足度を評価するため，利用開始の即時性を評価する．具体的には，利用開始時に即利用開始できた割合である．

3.2 閾値の設定

閾値の設定を算術平均とし，基本的な分析を行なう．以下に，閾値の計算式を示す．なお，m はドライバの数，n は現在から過去までの記録した情報の数である．

(1)　$H_{th} = 0$

(2)　$S_{th} = \frac{1}{m}\sum_{j=1}^{m} S_j$，　$S_a = \frac{1}{n}\sum_{i=1}^{n} S_{i,a}$

($S_{i,a}$: i 回前のドライバ a の利用開始時間)

(3)　$U_{th} = \frac{1}{m}\sum_{j=1}^{m} U_j$

(U_j:ドライバ j の Ut 時間内の利用回数)

(4)　$N_{th} = \frac{1}{m}\sum_{j=1}^{m} N_j$

(N_j:ドライバ j の最後に利用終了してから現時刻までの 時間)

履歴収集期間の閾値 (H_{th}) は，動作履歴を最大限利用するため，0 とする．利用開始時間の閾値 (S_{th}) は，過去 n 回の計算機の動作の中での全ドライバのデバイス検出から最初の利用開始までの時間の平均である．利用度の閾値 (U_{th}) は，全ドライバの Ut 時間内の利用開始の回数の平均である．非利用時間の閾値 (N_{th}) は，利用終了したドライバの現時刻までの経過時間の平均である．

表 2　シミュレーションに用いるドライバの情報

名前	大きさ (KB)	読込み時間 (ms) ※大きさより計算	初期化時間 (ms)
ドライバ 0	22.30	11.25	0.01
ドライバ 1	36.62	16.27	998.76
ドライバ 2	4.62	5.07	109.86
ドライバ 3	20.55	10.64	0.31
ドライバ 4	23.52	11.68	0.01
ドライバ 5	10.43	7.10	0.01
ドライバ 6	5.77	5.47	109.87
ドライバ 7	303.35	109.62	0.02
ドライバ 8	15.05	8.72	0.23
ドライバ 9	19.73	10.36	0.08

4. シミュレーション評価

4.1 シミュレーションの設定

制御法のソフトウェアシミュレータを作成し，種々の利用パターンに対する制御の有効性を明らかにする．

利用するドライバ 10 個の情報を表 2 に示す．プログラムの大きさは，FreeBSD4.3R の LKM のドライバから選択した．読み込み時間と初期化時間は，CPU：Intel Pentium4 2.8GHz，メモリ：512MB の計算機上での実測時間である．なお，すべてのドライバが制御する入出力機器は，立ち上がり時に接続されていると仮定する．また，一回のシミュレーションで計算機の起動を 15 回繰り返し，一回の計算機の起動で 8 時間の動作を仮定する．

4.2 利用開始時間の変化に対する分析

利用者の作業内容の変化により，入出力機器の使い方は変化する．つまり，利用パターンが変化する．ここでは，利用開始時間が変化する場合について，シミュレーション結果を述べる．利用パターンを表 3 に示す．すべての利用パターンは，均等な間隔で利用開始と利用終了を 10 回くり返す．パターン 0 から 3 は，パターン 4 から 7 より 500 秒遅く入出力機器の利用を開始する．これらの利用パターンを用い，以下の 4 つの変化について評価する．

(1)　利用パターンを変化させない場合
10 個のドライバのうち 4 つのドライバは，常に，パターン 0 から 3 までの 4 つのパターンで利用する．

(2)　利用パターンを途中で一度変化させる場合
6 回目の起動時までは，(1) と同じ利用パターンを用い，6 回目以降，パターン 4 から 7 を用いる．

(3)　途中で一時的に変化する場合
6 回目の起動時以外は，(1) と同じ利用パターンを用い，6 回目の動作のみ，パターン 4 から 7 を用いる．

(4)　利用するドライバが変化する場合
6 回目の起動時までは，(1) と同じ利用パターンを用い，6 回目以降，別の 4 つのドライバをパターン 0 から 3 で利用する．

各場合の立ち上がり時間と占有メモリ量の変化を図 2，図 3，図 4，および図 5 に示す．また，即利用開始できた割合を図 6 に示す．各図より，以下のことがわかる．

表 3　利用開始時間を変化させた利用パターン

利用パターン名	利用開始時間 (s)	利用回数
パターン 0	600	10
パターン 1	800	10
パターン 2	1000	10
パターン 3	1200	10
パターン 4	100	10
パターン 5	300	10
パターン 6	500	10
パターン 7	700	10

(1)　図 2 から図 5 まで，すべてに共通して，2 回目の起動以降，占有メモリ量，および立ち上がり時間は，低下している．これは，次の理由による．最初の起動において，動作履歴の無いドライバは，すべて即時起動を行う．この時，同時に動作履歴の収集を開始する．このため，一度利用されると次の計算機の動作からは，その時の動作に合わせ制御を行う．このことから，初期状態からの適応は早いといえる．

(2)　図 3 より，7 回目の起動から，立ち上がり時間は，段階的に変化している．利用パターンを途中で一度変化させる場合，全体的な利用開始時間は，変化する．これにより，立ち上がり時に起動するドライバの数は一時的に大きく変化する．その後，適応していく．また，n が大きいとき，変化は遅い．

(3)　図 4 より，n が大きいとき，立ち上がり時間は，利用開始時間の変化による影響を受けにくくなっている．

(4)　図 5 より，6 回目の起動から，最大占有メモリ量，および立ち上がり時間は，大きく変化している．これは，占有メモリ量と立ち上がり時間がドライバの種類の違いに大きく影響を受けるためである．そして，6 回目の起動時には，他の回の起動時と比較し，多くのドライバを読み込む．これは，立ち上がり時に利用しないドライバを読み込み，利用するドライバを利用開始の要求時に読み込むためである．

(5)　図 5 より，最大占有メモリ量，および立ち上がり時間は，n の変化に影響を受けていない．これは，次の理由による．閾値の計算方法は，ドライバの種類の違いを考慮していてない．また，利用開始時間の変化に比べドライバの読み込みと初期化の時間が非常に小さい．このため，n の変化に影響を受けていない．実際の計算機の利用においても，利用開始時間の変化に比べドライバの読み込みと初期化の時間は，小さいと考えられる．

(6)　図 6 より，6 回目の起動時には即利用開始できた確率は大きく低下している．これは，変化する以前に利用していなかったドライバを利用するためである．

4.3 利用時刻の偏りに対する分析

ここでは，利用する時刻に偏りのある利用パターンを用いた場合について，シミュレーション結果を述べる．利用パターンを表 4 に示す．各利用パターンは，偏った時刻に利用開始と利用終了を 10 回くり返す．パターン 8 は，均等な間隔で利用している．パターン 9 は，立ち上がり時と終了時に利用が集中している．パターン 10 は，立ち上がり時に利用が集中している．パターン 11 は，終了時に利用が集中している．

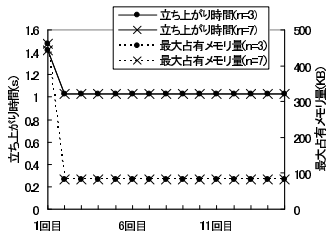


図 2　利用パターンが変化しない場合の制御

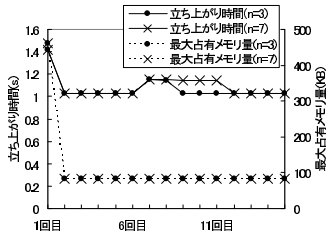


図 3　利用パターンが途中で一度変化する場合の制御

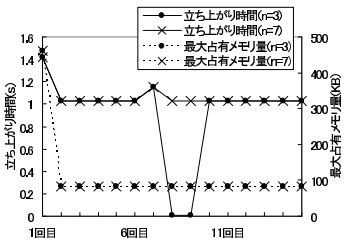


図 4　利用パターンが一時的に変化する場合の制御

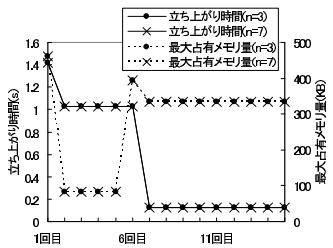


図 5　利用するドライバが途中で一度変化する場合の制御

10 個のうち 4 つのドライバは，常に，パターン 8 から 11 までの 4 つのパターンで利用する．この場合の占有メモリ量の変化を図 7 に示す．図 7 より，以下のことがわかる．

(1)　起動回数に関係なく，Ut が 4 時間の時，占有メモリ量は，5 時間経過した時点で低下している．これは，立ち上がり時に利用していたドライバを自動削除したためである．これに対し，Ut が 16 時間の時，自動削除は行われない．したがって，自動削除は，Ut を計算機一回の動作時間より短くした時に起こり得る．

(2)　2 回目以降の場合，占有メモリ量は減っている．これは，利用されないドライバの起動が抑制されるためである．

(3)　2 回目以降の場合，途中で占有メモリ量が増加している．これは，遅延起動のためである．

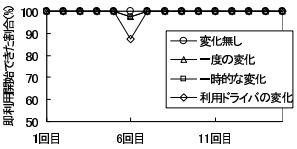
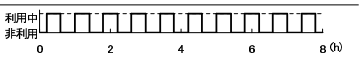
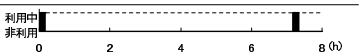
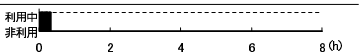
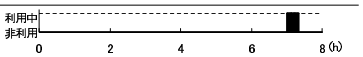


図 6　即利用開始できた割合の変化

表 4　利用時刻に偏りのある利用パターン

パターン名	利用パターン
パターン 8	
パターン 9	
パターン 10	
パターン 11	

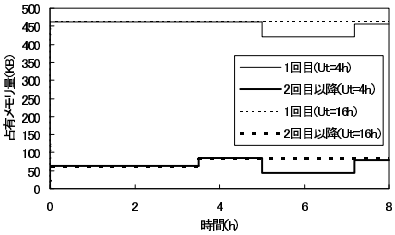


図 7　利用時刻に偏りのある利用パターンでの制御

5.　*AnT* による評価

5.1　*AnT* オペレーティングシステム

AnT オペレーティングシステム [2](An operating system with Adaptability and Toughness) は，プログラム構造としてマイクロカーネル構造を採用し，ドライバやファイルシステムといった機能をプロセスとして動作させる．プログラムは，OS とサービスからなる．OS は，内コアとプロセスとして動作する外コアからなる．サービスは，利用者の要求するサービスを提供するプロセスからなる．内コアは，システムの最小限な機能の動作を保証するプログラム部分である．主な機能として，メモリ領域管理，プロセスの実行制御部がある．外コアは，適応したシステムに必須なプログラムであり，動的に再構成可能な構造を有する．主な機能として，デバイスドライバやファイル管理がある．サービスは，利用者の要求する機能を提供する

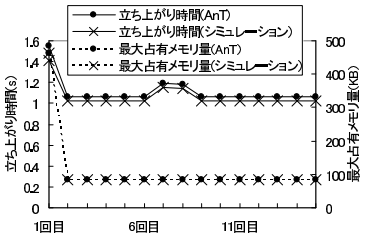


図 8 ***AnT*** での制御とシミュレーションでの制御の比較

部分である．

5.2 制御評価

制御法を ***AnT*** オペレーティングシステムへ実現し，先のシミュレーションの結果と比較する．具体的には，図 3 における n=3 の場合と同じ利用開始時間の変化について比較する．評価に用いた計算機は，CPU : Intel Pentium4 2.8GHz，メモリ : 512MB である．表 2 に示すドライバの情報に基づき，擬似ドライバプロセスを作成した．なお，初期化処理は，スピンループによる CPU 処理とした．

実測結果を図 8 に示す．最大占有メモリ量と立ち上がり時間は，シミュレーション結果 (図 3) と同様になっている．したがって，***AnT*** においてもシミュレーションと同様に制御できているといえる．

6. 占有メモリ量の比較

6.1 ドライバの実現方式とプログラムの大きさ

ドライバの実現方式として，OS 一体型ドライバ，LKM ドライバ，およびプロセス化ドライバがある．それぞれの特徴とプログラムの大きさの予想を表 5 に示す．OS 一体型ドライバは，モノリシックカーネルに組み込まれたドライバである．LKM ドライバは，動的リンクにより OS 走行中に組み込み取り外しが可能なドライバである．プロセス化ドライバは，プロセスとして走行するドライバである．LKM ドライバは，OS 一体型ドライバに比べ，動的リンクのために，再配置情報とシンボル情報を持つ．プロセス化ドライバは，OS 一体型ドライバや LKM ドライバに比べ，カーネルの機能を直接利用することはできない．したがって，プログラムの大きさは，「OS 一体型ドライバ < LKM ドライバ<プロセス化ドライバ」と考えられる．なお，以降では，各実現方式のプログラムの大きさを比較し，占有メモリ量の推察とする．

6.2 ドライバの LKM 化

FreeBSD4.3R の 139 個の LKM について，NIC ドライバ 5 個，USB ドライバ 5 個，および RAID コントローラ 5 個を対象にドライバを LKM 化した場合のプログラムの大きさを比較する．比較を図 9 に示す．なお，図 9 は，ドライバの大きさの平均である．平均して約 8.6KB 増加している．セクション別の大きさを図 10 に示す．LKM 化により，再配置情報とシンボル情報が増加し，全体の 78%を占めている．

表 5 各実現方式の特徴

実現方式	特徴	プログラムの大きさ
OS 一体型ドライバ	(1) OS 構築時に組み込み取り外しが可能 (2) カーネルの機能を直接利用できる (3) 動的リンク用の情報を持たない	小
LKM ドライバ	(1) OS 走行中に組み込み取り外しが可能 (2) カーネルの機能を直接利用できる (3) 動的リンク用の情報を持つ	
プロセス化ドライバ	(1) OS 走行中に組み込み取り外しが可能 (2) カーネルの機能を直接利用できない (3) 動的リンク用の情報を持たない	大

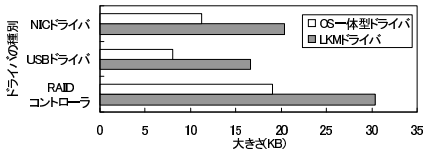


図 9 LKM 化時のプログラムの大きさ

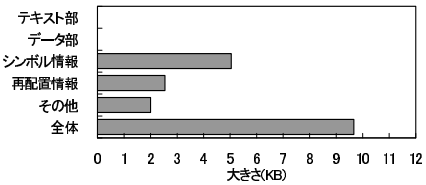


図 10 LKM 化時のセクション別の大きさ

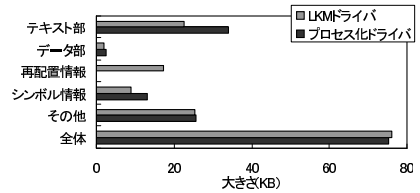


図 11 FD ドライバのプロセス化時のプログラムの大きさ

6.3 ドライバのプロセス化

既存の LKM ドライバを ***AnT*** 上でプロセス化する手法を提案した [3]．文献 [3] では，フロッピーディスク (FD) の LKM ドライバをプロセス化した．このドライバ (FD ドライバ) を対象に，ドライバをプロセス化した場合のプログラムの大きさを比較する．

比較を図 11 に示す．全体の大きさは，プロセス化ドライバが 0.95KB 小さくなっている．これは，次の理由による．テキスト部とシンボル情報は，合計 16KB 増加している．これは，プロセス化のために必要となった処理 (主に，プロセス間通信処理やワークキュー機能に関連する処理) のためである．一方，再配置情報は，動的リンクを行わないプロセス化ドライバには

必要ない．したがって，ドライバをプロセス化してもプログラムの大きさは減少する．

以上のことから，LKM ドライバをプロセス化する場合のプログラムの大きさの増減は，プロセス化のために必要となる処理の増加と再位置情報の削除の大小関係により決定される．

7. 関 連 研 究

計算機の立ち上がり時，ドライバの処理を工夫し，その処理時間を短縮する種々の手法がある [4]．例えば，デバイス認識処理を省略する手法，デバイス初期化処理時の I/O 待ち時間を最適な時間まで短縮する手法がある．これらの手法は，ハードウェアに依存するため，適応領域を限定されてしまう．これに対し，本制御法は，利用に基づく入出力機器の動作履歴を基に制御しているため，ハードウェアに依存しない．

ドライバの初期化処理では，I/O 待ちが発生する．この I/O 待ちの時間を有効利用するため，ドライバの初期化処理の並列化は有効である [5]．制御法を実現した ***AnT*** においても，ドライバはプロセスとして走行するため，初期化処理は並列化され，立ち上がり時間を短縮できる．

計算機の立ち上がり時間を短縮するため，立ち上がり時に必要となるプログラムを ROM に配置し，立ち上がり時のプログラム読み込み処理を省略し，直接実行する Execute-In-Place(XIP) という技術がある．このため，XIP では，開発段階で ROM に配置するプログラムを決定する必要がある．この決定手法として，利用パターンを事前に分析して利用する方法が提案されている [6]．これに対し，本制御法は，利用に基づく入出力機器の動作履歴を基に制御しているため，利用パターンの変化に追従できる特徴がある．

8. お わ り に

入出力機器の動作履歴を考慮し，ドライバプログラムを起動する制御法について，種々の利用パターンに対する制御の有効性を明らかにした．評価では，閾値に算術平均を用い，基本的な分析を行った．

シミュレーション評価により，初期状態からの適応は早いことを明らかにした．また，自動削除は，利用回数を記録しておく期間を一回の計算機の動作時間より短くした時に起こり得ることを明らかにした．次に，制御法を ***AnT*** オペレーティングシステムへ実現し評価した．その結果，シミュレーションと同様な結果を得た．

最後に，占有メモリ量を推察するため，プログラムの大きさを比較した．LKM ドライバとプロセス化ドライバは，OS 一体型ドライバに比べ，プログラムが大きい．また，LKM ドライバをプロセス化する場合のプログラムの大きさの増減は，プロセス化のために必要となる処理の増加と再配置情報の削除の大小関係により決定される．

残された課題として，より詳細な評価がある．

謝辞 本研究の一部は，科学研究費補助金 基盤研究 (B)「適応性と頑健性を有する基盤ソフトウェアのカーネル開発」(課題番号:18300010) による．

文　　献

[1] 滝口真一，谷口秀夫，“***AnT*** における入出力機器動作履歴を考慮したドライバプログラム起動制御法，”電子情報通信学会技術研究報告，Vol.107，No.558，pp.43-48，Apr. 2008.

[2] 谷口秀夫，乃村能成，田端利宏，安達俊光，野村裕佑，梅本昌典，仁科匡人，“適応性と堅牢性をあわせ持つ ***AnT*** オペレーティングシステム，”情報処理学会研究報告，2006-OS-103，Vol.2006，No.86，pp.71-78，Jul. 2006.

[3] 島崎 泰, 谷口 秀夫, 田端 利宏, 乃村 能成，“***AnT*** オペレーティングシステムにおける Linux の FD ドライバのプロセス化手法，”FIT2008 第 7 回情報科学技術フォーラム 講演論文集 第 1 分冊, pp.179-180, Sep. 2008.

[4] Tim R. Bird, ”Methods to Improve Bootup Time in Linux,” Linux Symposium 2004, Vol.1, pp.79-88, Jul. 2004.

[5] ”Linux Bootup Time Reduction for Digital Still Camera,” Chanju Park, Kyuhyung Kim, Youngjun Jang, Kyungju Hyun, Linux Symposium 2004, Vol.2, pp.231-340, Jul. 2006

[6] Tony Benavides, Justin Treon, Jared Hulbert and Weide Chang, ”The Enabling of an Execute-In-Place Architecture to Reduce the Embedded System Memory Footprint and Boot Time,”Journal of Computers, Vol.3, No.1, pp.79-89, Jan. 2008.