走行中のプロセスを
他プロセスから複製する手法の実現

平成29年2月15日

岡山大学 工学部 情報系学科

鈴木 森羅

研究背景

＜UNIXオペレーティングシステムにおけるプロセス複製＞

fork()システムコールにより自プロセスを複製

問題点: 走行中のプロセスを他プロセスから複製不可能

対象がfork()システムコールを発行するように改変

⇔ 他プロセスから複製できれば，改変不要

↓

走行中のプロセスを他プロセスから複製する手法を実現

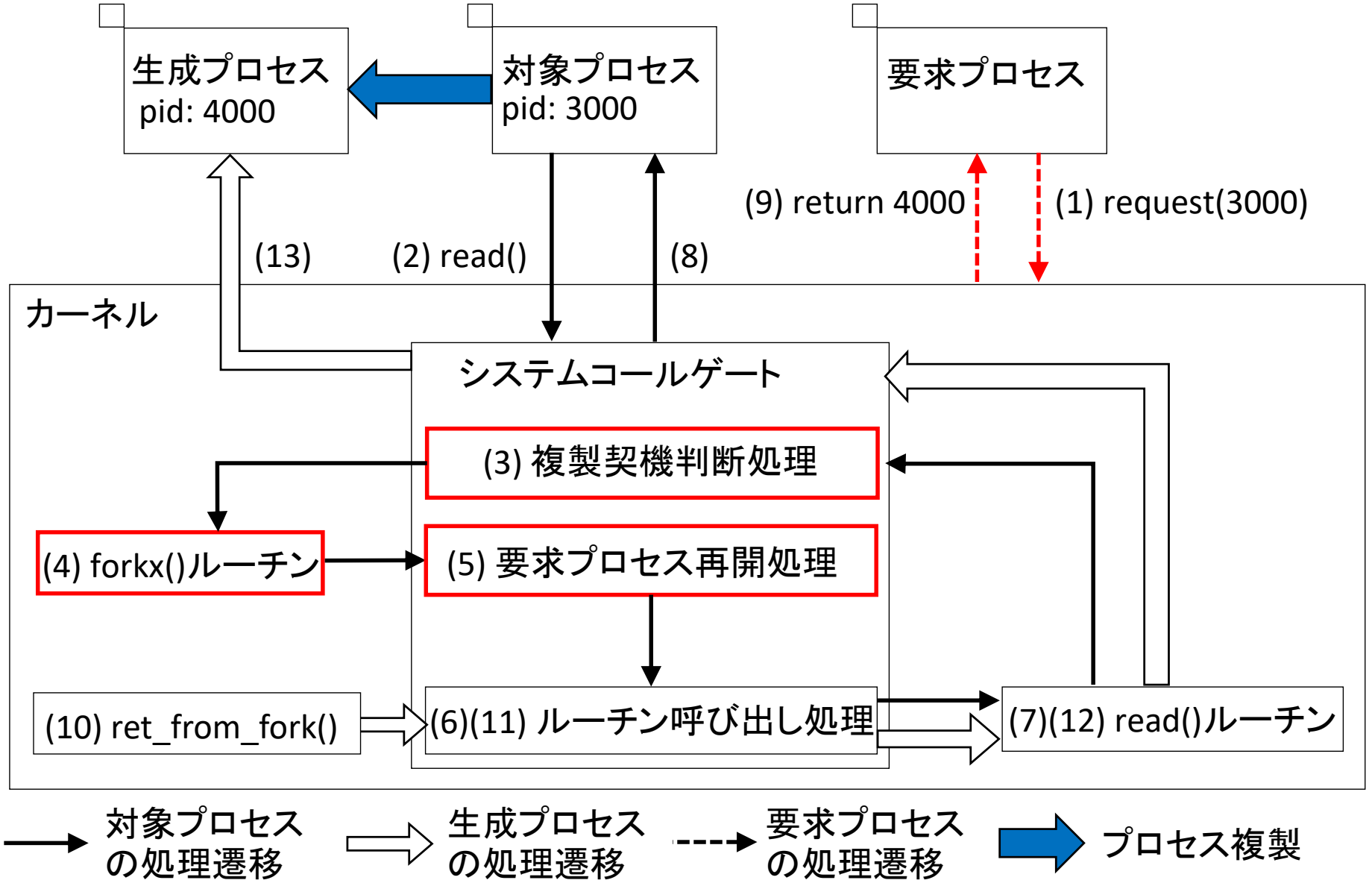
(1) 他プロセスからカーネルへ複製を要求

(2) 複製対象が最初に発行したシステムコールを契機に複製

∵ システムコールは多くのプロセスで発行される可能性高

(3) システムコールの処理中にfork()ルーチンを実行し，複製

処理流れ


生成プロセス
pid: 4000
対象プロセス
pid: 3000
要求プロセス
(9) return 4000
(1) request(3000)
(13)
(2) read()
(8)
カーネル
システムコールゲート
(3) 複製契機判断処理
(4) forkx()ルーチン
(5) 要求プロセス再開処理
(10) ret_from_fork()
(6)(11) ルーチン呼び出し処理
(7)(12) read()ルーチン
対象プロセス
の処理遷移
生成プロセス
の処理遷移
要求プロセス
の処理遷移
プロセス複製

評価

＜環境＞

OS	Debian 7.10
カーネル	Linux 3.0.8 64bit
CPU	Intel® Core™ i7,　2.93GHz
メモリ	2GB

＜観点＞

(1) fork()システムコールによる複製との性能比較

(2) システムコールにかかるオーバヘッド

(3) 要求プロセスの待機時間

(4) 提案手法の実装によるコード改変量

fork()システムコールによる複製との性能比較

性能を比較し，提案手法による複製の処理時間を評価

結果: 同程度の処理時間で複製

複製の処理時間:

複製方法	処理時間(μs)
提案手法	41.12
fork()システムコール	40.27

差は0.85μs(約2%)であり，小

システムコールにかかるオーバヘッド

全てのプロセスのシステムコールを契機判断の必要有

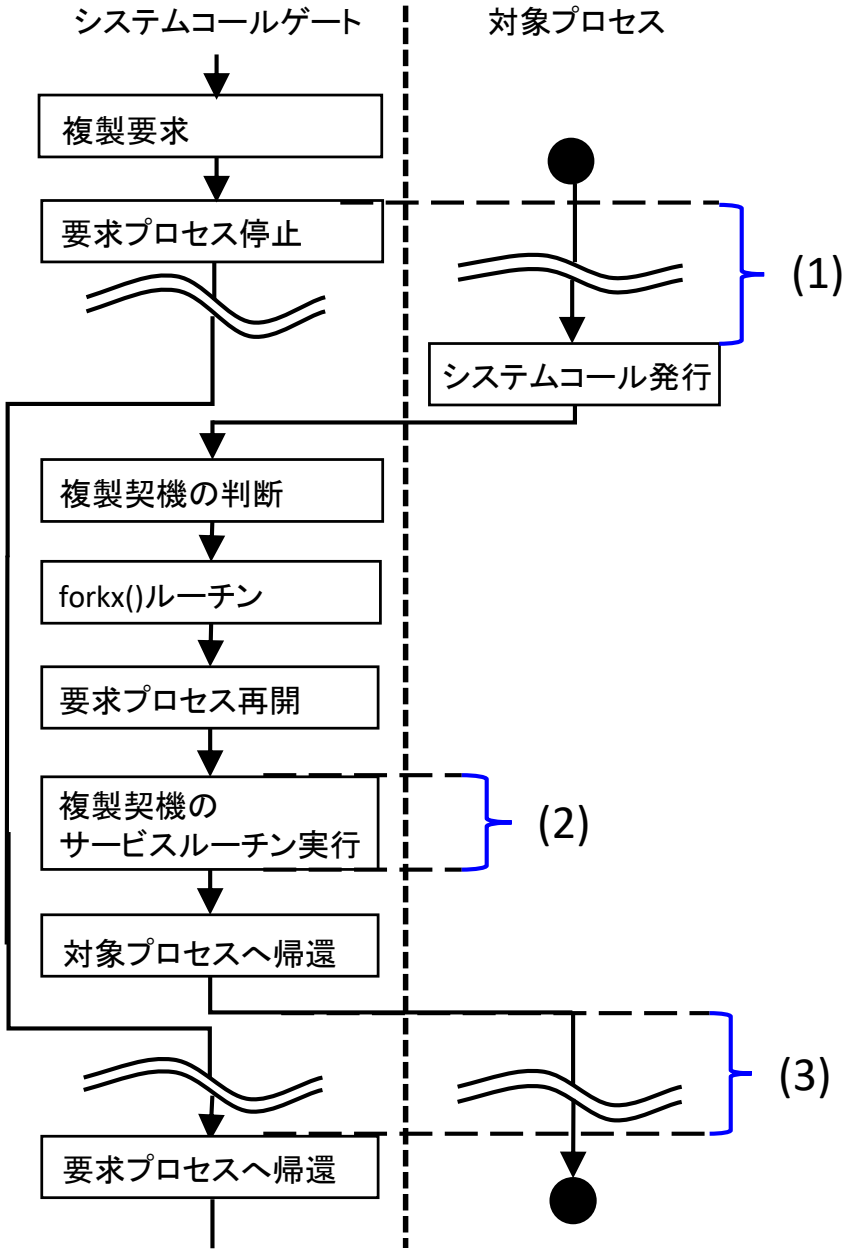
∴ オーバヘッド発生

getpid()システムコールの場合:

カーネル	処理時間(μs)
改変前	0.74
改変後	0.78

結果: オーバヘッドは0.04μs(約6%)であり，小

要求プロセスの待機時間



対象や契機によって処理時間が一定ではない処理有

一定ではない要因

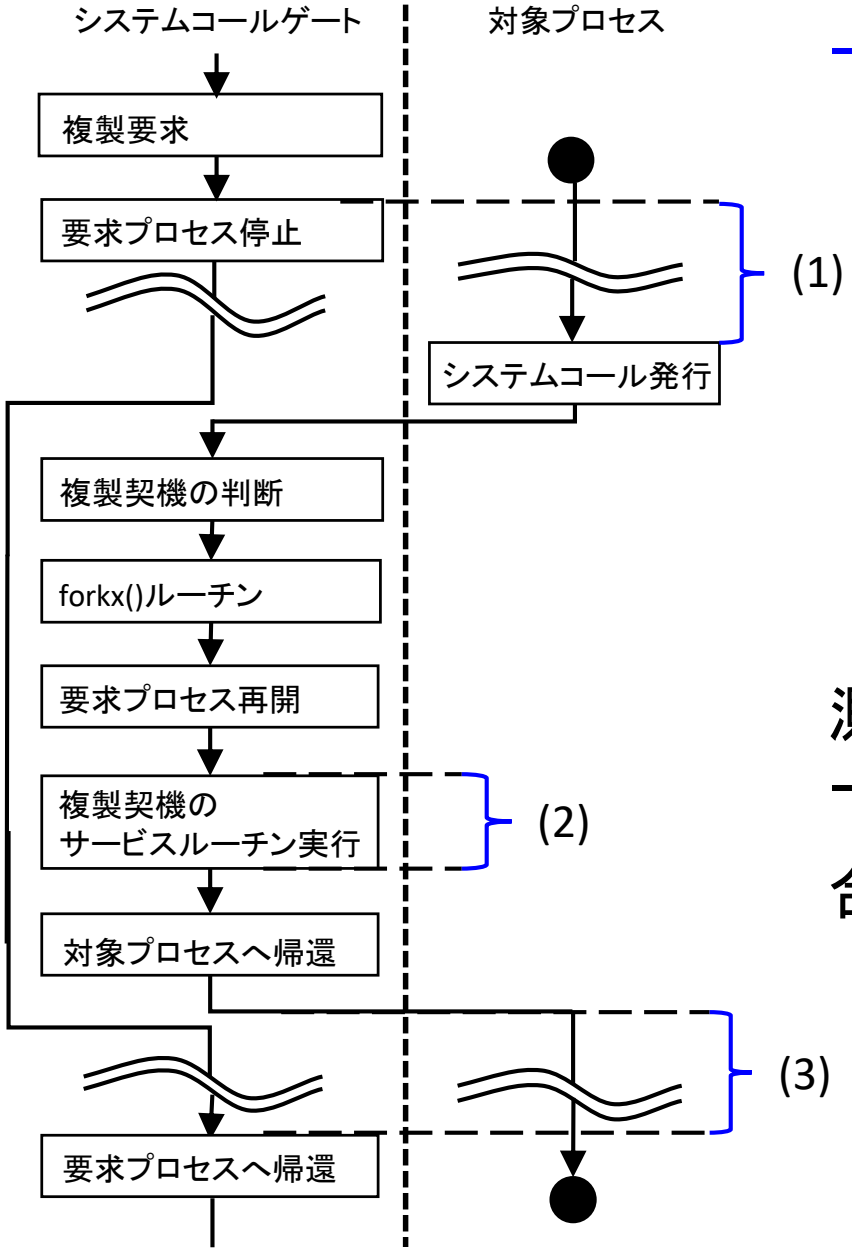
(1) 対象のシステムコールの発行間隔や要求のタイミング

(2) 複製契機のシステムコールによる処理内容の差異

(3) プロセスのスケジューリング状況

これらの要因の影響を評価

評価の結果



一般的な例(対象: vim，契機: getpid())

(1) 506.22μs: vimのシステムコール平均発行間隔の半分

(2) 0.20μs: getpid()のサービスルーチンの実行時間

(3) 68.60μs: 評価環境での測定における平均時間

測定の結果，
一定な処理時間の合計は44.97μs

合計619.99μs，うち約82%が(1)

∴ 対象のシステムコール平均発行間隔の影響大

提案手法の実装によるコード改変量

(1) 実装による改変箇所
  (A) 複製要求システムコール
  (B) システムコールゲート
  (C) forkx()ルーチン
  (D) ret_from_fork()
(2) 全6ファイルに渡って，55行のコードを追加
(3) ファイル数，コード量ともにLinuxカーネルに比べ，小

∴ 提案手法の導入は容易

まとめ

＜実績＞

(1) 走行中のプロセスを他プロセスから複製する手法を実現

(A) 対象プロセスのシステムコール発行を契機に複製

(2) 提案手法を評価

(A) 既存のプロセス複製と同程度の時間で複製

(B) 手法の導入が容易

＜今後の課題＞

(1) 生成プロセスの入出力対象の変更

(2) 複数のプロセスからなるアプリケーションの複製への対応

予備スライド

fork()について

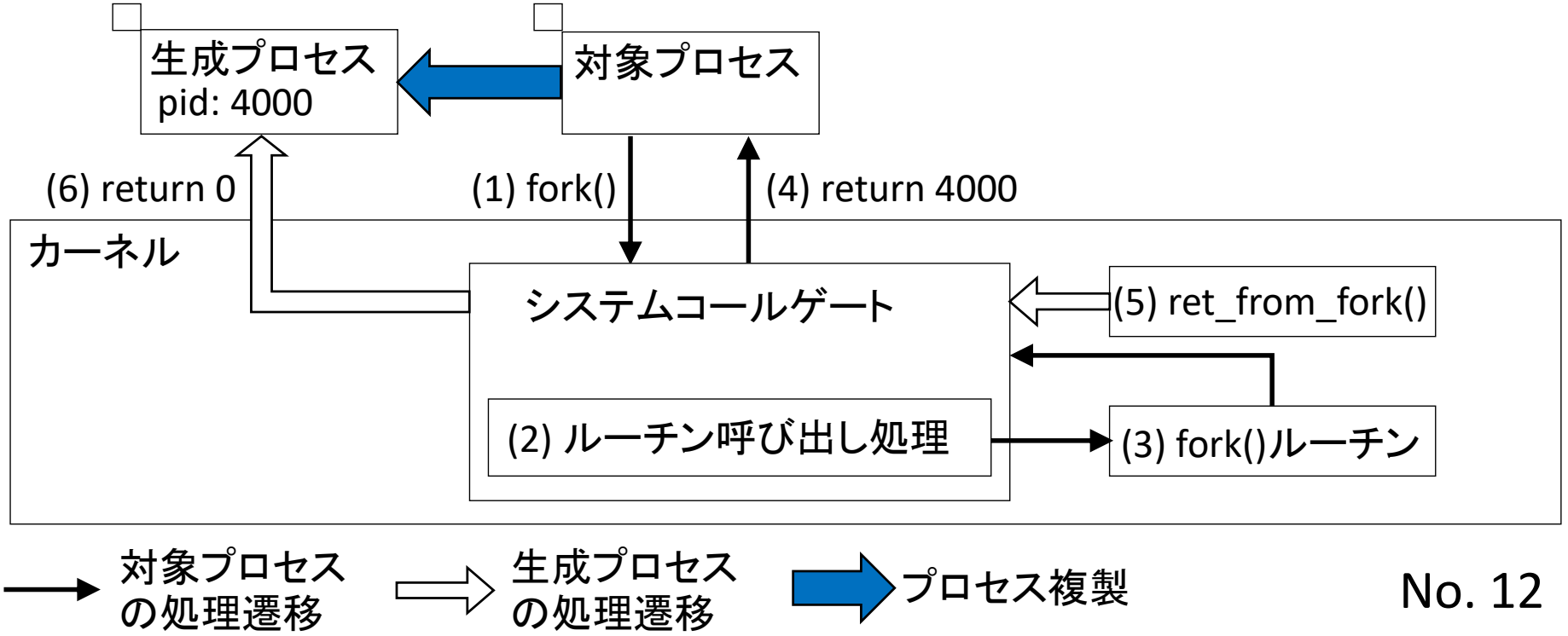
＜fork()システムコール＞

(1) 自プロセスのみを複製可能

(2) 親プロセスへの返り値: 子プロセスのPID

　　子プロセスへの返り値: 0

(3) 生成プロセスはret_from_fork()から処理を開始



要件と方針

要件1: 要求プロセスが対象プロセスを指定可能

対処1: 要求プロセスはPIDを用いて対象プロセスを指定

要件2: 要求プロセスが任意の時点で複製を要求可能

対処2: 要求プロセスはシステムコールを用いて複製を要求

要件3: 対象プロセスのソースコードを改変不要

対処3: 対象プロセスの発行したシステムコールを契機に複製

要件4: 導入工数の削減

対処4: システムコールゲートを改変

要件5: 対象プロセスと生成プロセスが独立に走行

対処5:対象プロセスと生成プロセスが独立となる複製

(A) 親子関係の解消

(B) 入出力対象の変更

課題と対処

課題1: 要求プロセスの停止/再開

　対処1: プロセスの状態を変更

課題2: forkx()ルーチンにより書き換えられたレジスタの復元

　対処2: カーネルスタックに保存された値を基に復元

課題3: 複製契機のサービスルーチンの実行

　対処3: ret_from_fork()の処理遷移先の変更

課題4: 親子関係を結ばない複製

　対処4: フラグの設定により親プロセスを共有

論理LOC

論理LOC

空白行，コメントのみの行，および記号のみの行を除くコード行数

	ファイル数	論理LOC
Linux 3.0.8	32701	9918509
提案手法の実装による改変	6	55

提案手法の実装による改変の全体に対する割合

(1) ファイル数: 約0.02%

(2) 論理LOC:　約0.0005%

提案手法の実装による改変の詳細

改変箇所 / ファイル名	複製要求システムコール	システムコールゲート	forkx()ルーチン	ret_from_fork()	合計
/arch/x86/include/asm/syscalls.h	2	-	-	-	2
/arch/x86/include/asm/unisted_64.h	4	-	-	-	4
/arch/x86/kernel/Makefile	1	-	-	-	1
/arch/x86/kernel/entry_64.S	-	23	-	7	30
/arch/x86/kernel/process.c	-	-	2	-	2
/arch/x86/kernel/sys_request_fork.c	16	-	-	-	16
合計	23	23	2	7	55