プロセスの実行イメージの履歴の獲得についての研究

舩津 俊一 （最所研究室）

あらまし

本研究においては、LINUX を対象としたプロセスの実行イメージの履歴の獲得についての研究を行った。LINUX の場合、プロセスの完全な実行状態を表している実行イメージを、ptrace システムコールを用いることによって獲得できる。コピーオンライトの手法を用いることによって変更箇所の絞り込みを行った。さらに、取り出した実行イメージは、差分情報を、圧縮することで、データサイズを小さくした。

1 はじめに

現在、世界中の計算機はネットワークでつながっていると言っても過言ではない。毎日のように、新種のコンピュータウィルスが出現し、クラッカーによる違法アクセスも日常茶飯事のように行われているのが現状である。そのためにワクチンプログラムやファイヤーウォールによる対策を施しているが、現在のワクチンプログラムは、ファイルに感染したウィルスに対してのものであるため、メモリーを直接書き換えるタイプには対応できず、プロセスの終了とともに消えるので痕跡が残りにくい。このような場合の痕跡を見つけるのに、プロセスの実行イメージの履歴が役に立つと考えられる。

また、ソフトウエア開発におけるデバッグにも、プロセスの実行イメージの履歴が有用であると考えられる。バグというものは、プログラムの規模が大きくなればなるほど、また複雑なものほど取り除くのがやっかいである。開発過程の中で、デバッグやテストにかかる時間は相当なものであり、開発期間の半分以上を使っていることも珍しいことではない。プロセスの実行イメージの履歴は、バグの発生した状況を容易に再現することができる可能性を秘めている。

プロセス実行イメージの履歴は、ネットワーク管理やソフトウエア開発のような高度な専門的作業だけでなく、一般のオフィスなどの事務作業にも活用できる。もし、報告書を作成している途中に、計算機がなんらかの理由で、意図しない停止を起こした場合、それまでのユーザの作業は無駄になってしまう。作業をもう一度やり直すのは、非常に手間が掛かる作業である。しかし、プロセスの実行イメージの履歴があれば、それを元に停止前の状態まで計算機の状態を戻すことも可能な場合も考えられる。

以上のことから、プロセスの実行イメージは、非常に有益なものであると考え、プロセスの実行履歴を獲得するシステムの設計を行い、実行履歴の管理のオーバーヘッドを評価した。

2 プロセス実行イメージの獲得

LINUX においては、/proc/<pid>/maps 情報から、セグメントの情報を読み取り、前回の maps 情報と比較して変更が起きたセグメントを ptrace システムコール [1] を用いて変更されたメモリー空間を獲得できる。ここで、変更が起きたセグメントの特定が可能なのは、コピーオンライトを呼ばれる手法を用いているからである。コピーオンライトでは、変更前のデータと変更後のデータが残っており、さらに、変更した箇所を特定できる。その様子を図 1 に示す。

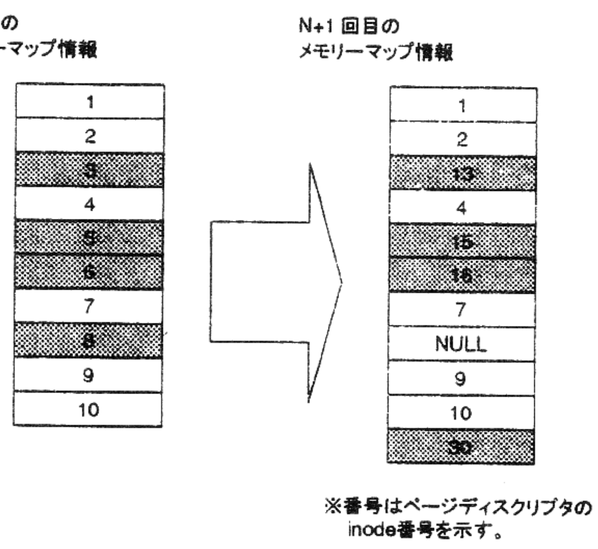

図 1: メモリーマップ情報の変化

LINUX においては、仮想記憶はメモリーリージョンと呼ばれる単位で管理しており、図 1 の例では、3,5,6 番目のメモリーリージョンが 13,15,16 番目のメモリーリージョンにそれぞれコピーされ、8 番目のメモリーリージョンを使用しなくなり、30 番目のメモリーリージョンを新たに使用されていることを示している。

3 差分情報の生成

変更が起きたメモリーリージョン全てが前の状態と全て異なるわけではない。そこで、異なる部分（差分）だけを保存することにより、完全なデータからその地点までの差分情報を用いて、その地点のデータを再現

できる。

バイナリデータの差分として考えられるのは、ブロック単位での減算値と排他的論理和が、考えられる。前者は、ブロック単位（本研究では32bit単位とする）で減算を行い、その値を差分とするというものである。しかし、減算には交換法則が成り立たないので、差分に方向性が存在する。それに対して、排他的論理和の場合、減算値とは異なり交換法則が成立するし、新旧の両方向からの復元が可能になる。

本研究では、両方向からの復元が可能な排他的論理和を用いた。

4 プロセス実行イメージ履歴の管理

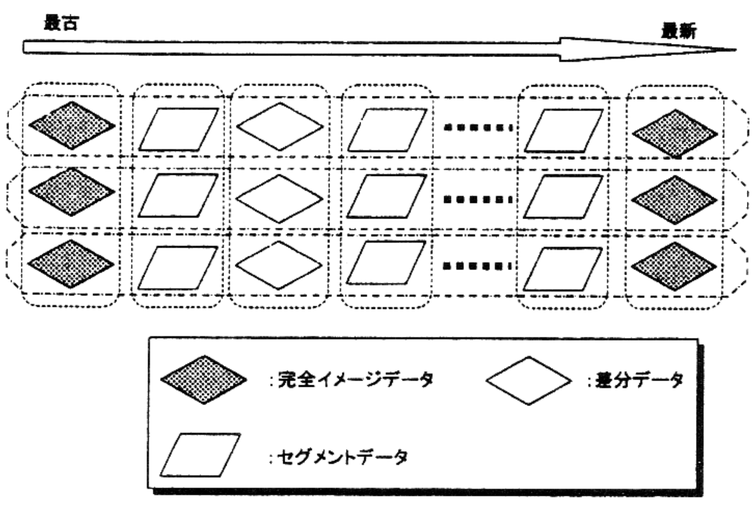

図 2: ファイルの保存方向

管理データは、図2で示す構造になっている。ファイルの作成法として、2種類が考えられる。1つのセグメントを1つのファイルとするセグメント単位と1回の獲得を1つのファイルとする獲得単位である。前者の場合、保存時に開くファイルの数は、システムから受け取る変更が起こったセグメントの数となる。それに対して、後者の場合、システムから受け取った1回の獲得で得られた、リージョンごとの差分データおよび、メモリーマップを保存するだけである。前者の方法では、いったんセグメント履歴ファイルの内容を読み、保存先のオフセットを特定する必要があり、効率が悪い。

よって、本研究では、後者の獲得ごとにファイルを作成する方法を採用した。

5 評価

オーバーヘッドの指標にTSC[2]というハードウェアクロックを用いたソフトウェアシミュレーションにより、このシステムの有効性について評価を行った。その結果、800MHzのPentiumIIIにおいて、50ms/MBとなり、対象プロセスに与えるオーバーヘッドは、よほど時間的制約が厳しいプロセスまたは、数十MByte以上の膨大なメモリー空間を使うプロセスでない限り、十分に実用的に使うことができるレベルであるという結果が得られた。

6 まとめ

以上、LINUXを対象としたプロセスの実行イメージの獲得について述べた。ptraceシステムコールを用いることによって、他のプロセスが使用しているメモリーイメージを獲得し、その差分により履歴を保存するシステムの設計を行った。

毎回プロセスの全メモリーイメージを獲得することは、オーバーヘッドが大きいので、コピーオンライトを用いることにより、変更が起きたリージョンを特定するという方法を用いた。

次に、排他的論理和を用いることによって、細かな差分を抽出し、さらにランレングス符号化を元にした圧縮を行うことによって、効率的にプロセスの実行イメージ履歴を管理する。

本論文で示した方法は多くの場合、対象プロセスに大きなオーバーヘッドをかけることなくプロセスの実行イメージの履歴を作成できる。履歴を獲得するオーバーヘッドを検証するために, ソフトウェアシミュレーションを行った。その結果、提案方法のオーバーヘッドは、800MHzのPentiumIIIにおいて、50ms/MBとなり、十分に実用に耐えるものであることが分かった。しかし、ptraceシステムコールは、実行イメージの獲得に利用するには、使い勝手がよいシステムコールとは言い難いものであり、別の方法を模索する必要がある。さらに、CVS[3]に代表されるファイルバージョン管理システムとの連携やプロセスの再実行などの課題が残っている。

謝辞

最所圭三教授には、研究および執筆にあたり数々の有益なご教示をいただきましたことに深甚の謝意を表します。

そして、研究活動はもとより論文作成にあたり、多大な協力いただきました多くの方々に厚くお礼申し上げます。

参考文献

[1] DANIEL P.BOVET, MARCO CESATI 著, 岡島潤治郎、田宮まや、三浦広志 訳, “詳解 LINUX カーネル”, O'REILLY (オーラリー・ジャパン) 2001

[2] “IA-32 インテル®アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル 中巻 命令セットリファレンス”, Intel Corporation 1997-2001, 資料番号：245471J

[3] 鯉江英隆、西本卓也、馬場肇 著, “バージョン管理システム（CVS）の導入と活用”, ソフトバンク パブリッシング 2000