プロセスの保存と復元によるGUIアプリケーションの起動高速化

阿部敏和† 中山泰一†

†電気通信大学 情報工学科

1 はじめに

近年、アプリケーションの多機能化に伴い起動時の初期化処理の計算量も増加傾向にある。その結果、多くのユーザがアプリケーションの起動に要する時間に不満を抱いている。

アプリケーション起動時の初期化処理の中には、動作環境や設定に変更が加えられない限り毎回同じ結果になる部分が多数存在する。そこで本研究では、初期化処理が終了した時点でのプロセスの情報を保存し、以後そのアプリケーションを起動する際には初期化処理が終了したプロセスを復元、再開することで初期化処理を省略し、起動高速化を図る。また、プロセスの保存を自動的に行うための初期化処理終了検出方法についても議論する。

最後に、既存のアプリケーションを用いて通常の起動方法と提案手法それぞれの実行時間を測定し、それらを比較することで提案手法の有効性を確認する。

2 関連技術

2.1 Checkpointing

Checkpointingは、実行中のプロセスを定期的に保存し、障害発生によりプロセスの処理継続が不可能となった際に最後に保存した時点から処理を再開する技術である。Linux上での実装例としてDMTCP[1]やBLCR[2]が存在する。

本来は、大規模な並列分散システムにおける耐障害性向上を目的として使用される技術であるが、本稿ではプロセスの保存にこのCheckpointingを利用する。

3 設計

3.1 アプリケーション起動時間短縮手法

アプリケーション起動時の初期化処理の中には、ライブラリのロードなどその動作結果が毎回同様となる処理が多数含まれている。アドオンやプラグインなどと呼ばれる拡張機能、動作設定が記述されたファイルの読み込みなども、拡張機能や設定内容への変更が行われない限り同じ結果となる。

そこで、初期化処理が終わった時点でのプロセスを保存して、それ以降の起動時には保存したプロセスを復元することで初期化処理の大部分を省略することが可能であると考えられる。本稿ではこの手法により起動時間の短縮を図る。

3.2 プロセス保存の自動化

本手法では実行中のアプリケーションプロセスをどのタイミングで保存するかが重要になる。しかし、起動時間の短縮に有効なタイミングでのプロセスの保存は、ユーザ自身が行うことが非常に困難である。

プロセスを保存するタイミングが早すぎると初期化処理中の省略される部分が少なくなる。

一方、プロセスを保存するタイミングが遅すぎると、保存したプロセスの汎用性が失われ、再利用が難しくなる。

そこで、本システムではアプリケーションの初期化処理終了時点を推定し、プロセスの保存を自動的に行う。初期化処理終了時点の推定にはシステムコールの呼び出しパターンを利用する。たとえば、socketcall()やmunmap()など特定のシステムコールが実行された直後や、ファイルパスの先頭が"/usr/lib"となっているopen()が一定回数呼ばれた後に最初にclose()が成功した直後などのようなパターンを予め定めておく。

4 実装

プロトタイプシステムの概要を図1に示す。

まず、ptrace()を使用して対象アプリケーションのプロセスによるシステムコールの呼び出しを全て監視する。そのシステムコールの呼び出しパターンから初期化完了を予想し、DMTCPを使用してそのプロセスを保存する。これにより、ユーザは特別な操作を行うことなく高速起動用のプロセスイメージを作成することができる。

Method for fast startup of GUI applications with saving and resuming state of the process.

Toshikazu ABE† and Yasuichi NAKAYAMA†

†Dept. of Computer Science, The University of Electro-Communications

abe-t@igo.cs.uec.ac.jp, yasu@cs.uec.ac.jp

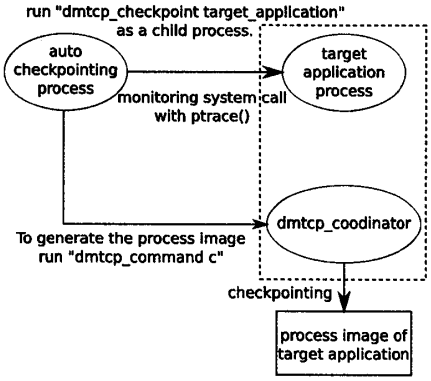

図 1: プロトタイプシステムの概要

表 1: 実行時間比較 (ディスクキャッシュ無効)

	no add-on	ECB
normal boot	2.518sec	4.963sec
DMTCP(auto)	2.616sec	5.315sec
DMTCP(manual)	2.217sec	2.681sec

5 評価

5.1 評価実験

提案手法の有効性を評価するため、通常の起動方法と提案手法それぞれについて実行時間を測定し、それらの比較を行った。

実験の対象としたアプリケーションは Emacs22.2.1 で、拡張機能をインストールしていない場合と、Emacs Code Browser をインストールした場合の 2 種類で実験を行った。初期化処理自動検出の条件は、簡単のため“標準出力に対する特定文字列の出力”とし、Emacs の設定ファイルである.emacs の最後に当該出力を行うように記述した。また、DMTCP を使用した起動方法については初期化処理完了をユーザが見届けてから手作業でプロセスの保存を行った場合についても測定を行った。

5.2 起動速度比較

表 1 はディスクキャッシュを無効にした場合、表 2 はディスクキャッシュを有効にした場合の結果である。

結果を見ると、Emacs の拡張機能の有無やディスクキャッシュの有無を問わず、ユーザによるプロセスの保存を行った場合の提案手法が最も速い。これは提案手法がアプリケーションの起動高速化に対して有効であることを示している。特にディスクキャッシュを有効にした場合や、対象アプリケーションが機能拡張などで大きくなるほど高速化の効果が高いことがわかる。しかし、プロセスの保存を自動化した場合は手動の場合よりもやや遅くなっている。これは初期化処理終了の判定基準が".emacs に記述された最後の処理が実行された時点"と非常に単純であることが原因として考えられる。今回の結果では多くの場合で通常起動よりも高速な起動を実現しているが、より有効な判定基準を検討する必要がある。

表 2: 実行時間比較 (ディスクキャッシュ有効)

	no add-on	ECB
normal boot	0.450sec	1.244sec
DMTCP(auto)	0.259sec	0.776sec
DMTCP(manual)	0.257sec	0.293sec

DMTCP の標準動作では保存したプロセスイメージを gzip 圧縮するが、本実装では圧縮ファイルの展開時間を省くためこれを無効にしている。この結果、圧縮した場合 [3] よりも高速に起動することができた。

6 まとめ

本稿では、プロセスの保存と復元を使用することによるアプリケーション起動時間の短縮手法と、その際のプロセスの自動保存手法の提案とその評価実験を行った。実験結果により、起動時間の短縮手法についてはその有効性が確認できた。一方、プロセスの自動保存についてはある程度の効果は確認できたがまだ改良の余地があると言える。

今後の課題としては、未だ対応できていない GUI アプリケーションへの対応などが挙げられる。

参考文献

[1] Jason Ansel, Kapil Arya, and Gene Cooperman: DMTCP: Transparent Checkpointing for Cluster Computations and the Desktop, *23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS'09)*, Rome, 2009, pp1-12.

[2] Paul H. Hargrove, Jason C. Duell: Berkeley Lab Checkpoint/Restart (BLCR) for Linux Clusters, *Journal of Physics: Conference Series*, 2006, v.46, pp494-499.

[3] 阿部敏和, 中山泰一: プロセスの保存と復元によるアプリケーション起動高速化手法, 日本ソフトウェア科学会第 26 回大会, 4B-2(2009).