継続の共有化による継続ベース Web サーバのメモリ使用量削減

新宮 澄夫 † 小宮 常康 † 佐藤 喬 † 多田 好克 †
電気通信大学 大学院情報システム学研究科 ‡

1. はじめに

従来の古典的な CGI による Web アプリケーションでは, ユーザとの対話のたびにプログラムの実行を一旦中断する. その際中断したプログラムを再開するための情報を保存し, 再開する際にはその情報を取り出す. 基本的にその情報の管理は開発者が記述しなければならず, 大きなアプリケーションにおいてはその管理が煩雑である. また管理コードが明示的に埋め込まれるため, そういった Web アプリケーションのプログラムは可読性が悪いものとなる.

そこで継続ベース Web サーバというものがある [1]. これは中断したプログラムの再開に必要な情報の保存, 復元に第一級継続を用いたもので,CUI によるコンソールアプリケーションを記述する場合と同様に, アプリケーションの処理の流れの通りプログラムを記述することが出来る. このように継続ベース Web サーバでは従来のスタイルの Web アプリケーションよりプログラムが可読性が良くなるという利点がある.

しかし継続ベース Web サーバの継続オブジェクトのサイズは従来の Web アプリケーションが残す情報のそれに比べ大きく, 継続ベース Web サーバは従来のものに比べより多くのメモリを消費する. 継続オブジェクトのサイズを小さくすることはサーバの負担軽減に繋がる. そこで本研究では, 継続の共有化による継続ベース Web サーバのメモリ使用量削減を提案する.

2. 基本的アイデア

スタックベースの処理系では継続を捕捉する関数 (call/cc) を呼ぶと, その時点での制御スタックの内容を継続オブジェクトとしてヒープに退避する. 継続を再開する時は退避した内容を言語処理系がヒープから制御スタックに戻し計算を再開する. プログラム中のある時点で捕捉した継続オブジェクトの内容は, その時点における制御スタックの内容である.

次のプログラムで作られる継続について考える.

```
(define (f)
  ... (call/cc ...) ...)
(define (g)
  ... (f) ...)
(g)
```

まず関数 g が呼ばれ制御スタックに g のフレームが積まれ, 同様に f のフレームが積まれる. 次に call/cc が呼ばれ, 制御スタックにある g と f のフレームが継続としてヒープにコピーされる. この call/cc が何度も呼ばれると,g と f のフレームからなる継続が何個も作られる. このような場合, 図 1 のように継続を一つにまとめて保存することでメモリを節約することが出来る.

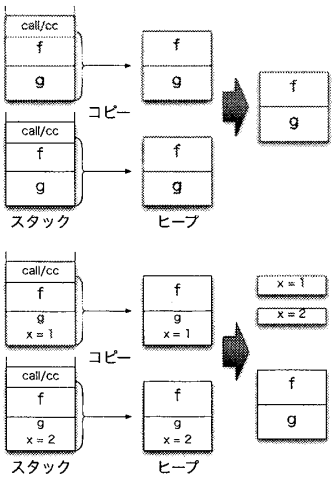

図 1: 共有の基本アイデア

また次のプログラムのようにフレームの並びが同じで局所変数が異なる場合がある.

```
(define (f)
  ... (call/cc ...) ...)
(define (g x)
  ... (f) ...)
(g 1)
(g 2)
```

このような場合でも局所変数だけ個別に保存して対応することができる.

継続ベース Web サーバに対して大量のユーザが同時に接続しているケースについて考える. それぞれのユーザとの対話のたびに継続が作られる, 大量の継続オブジェクトが同時に存在している. もしある call/cc が呼ばれる時のフレームの並びが常に同じならば, その call/cc で作られる継続については異なるセッション間で共有できる.

そこでプログラム中の各 call/cc について呼び出される際のフレームの並びを調べ, 常に同じ並びのものがあれば共有化を行う call/cc 関数へ置き換えることでメモリ使用量を削減できる.

3. 継続の共有化

3.1. 共有化の判断

プログラム中の異なる call/cc で作られる継続は明らかに共有できない. またプログラム中の同じ call/cc で作られる継続でも, そこへ至るまでの制御フローが複数存在することがある. その場合, 制御スタックのフレームの並びが異なるため共有化は困難である.

Reducing Memory Usage in Continuation-based Web Servers by Sharing Continuations
†Sumio Shingu, Tsuneyasu Komiya, Takashi Satou, Yoshikatsu Tada
‡Graduate School of Information Systems, The University of Electro-Communications

例えば関数 f から call/cc が, 関数 g と関数 h から f が呼ばれるとする. その場合 call/cc で作られる継続オブジェクトには g の関数フレームが含まれるものと h の関数フレームが含まれるものの二種類が考えられるため共有化できない. また f が g のみから,g が関数 g1 と関数 g2 から呼ばれる場合でも同様である. ただし g が g1 から呼ばれた時のみ f を呼び出すような場合,call/cc へ至る呼び出し順序は一つなので共有可能である.

継続の共有化を自動的に行うために, トップレベルから各 call/cc への制御フロー解析を行う. プログラム中の全ての call/cc について調べ, トップレベルからの制御フローが一つの call/cc についてのみ共有化を行う.

そこで Shivers の抽象解釈 [4] によるフロー解析を行った. 共有化する場面を多くするためフロー解析には g1 から呼ばれた g と,g2 から呼ばれた g を区別する精度が必要である. そのため 1CFA というフロー解析を使用する. これは同じ関数でもそこまでの呼び出し順序が異なる場合は区別して扱うため, 上記の要求を満たす.

3.2. 局所変数の解析

制御スタックの内容は大きく制御情報と局所変数の二つに分けられる. 制御情報はリターンアドレス等であり, 共有化する継続の場合制御情報は全く同一である. そのため制御情報については異なるセッションで作られる継続間で共有できる. しかし制御スタックの中で局所変数が占める割合は少なくない. そのため局所変数についても共有できる部分については共有したい.

局所変数は異なるセッション間で常に同じ値のものと, 異なる可能性があるものとの二種類に分けられ, 同じ値のものについては共有可能である. しかしある局所変数について共有できるか否かを, 継続を作るたびに共有している継続オブジェクトのものと比較して決めるのはコストを考えると望ましくない. そのためあらかじめ各局所変数について共有できるか否かを調べ, 共有できない局所変数を選び出しておく.

そこで 1CFA を拡張し, 代入などを調べることにより各局所変数について共有可能か否かの情報を集める.

3.3. 共有化した継続

継続オブジェクトをテンプレートと差分情報から構成する方法を提案する. 具体的には異なるセッションの継続間でテンプレートを共有し, 差分情報は個別に作り保持する. テンプレートには共通部分すなわち制御情報と共有可能な局所変数を, 差分情報には非共通部分すなわち共有できない局所変数を保存させる.

共有化する継続を作る時には局所変数についての情報に従い差分情報を作り, セッション間で共有しているテンプレートとその差分情報のセットを継続オブジェクトとする. 継続を再開する時はまずテンプレートに基づき共通部分を制御スタックに戻し, 次に差分情報に基づき非共通部分を制御スタックに戻す.

4. 評価

Scheme 処理系の一つである TUTScheme[3] に, 共有化した継続を捕捉する call/cc を実装した.

従来の call/cc と共有化した call/cc について, 継続ベース Web サーバの利用をシミュレーションしたプログラムでメモリ使用量を比較した. セッション数は 50000, 対話の数は 250000, 保存しておく継続の上限は 50000 とした. その結果を図 2 に示す. 縦軸は全体のメモリ使用量, 横軸はユーザとの対話の数である. またある程度メモリ使用量が大きくなると

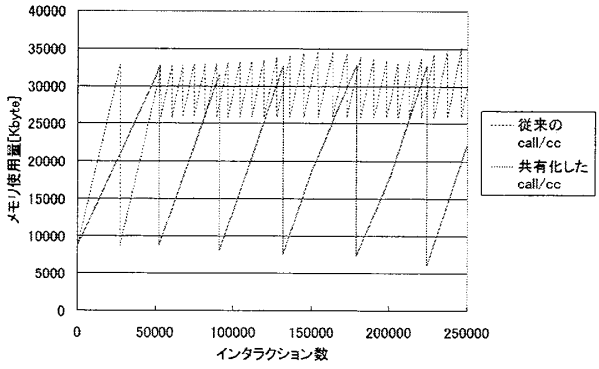

図 2: 継続ベース Web サーバにおけるメモリ使用量の比較

それが減少するのはガーベッジコレクション (以下 GC) による働きである.

従来法に比べ提案法は対話の発生に対してメモリ使用量の増加が少ない. また GC の頻度も少なく,GC 後に回収されずに残ったデータのサイズも小さい.

5. まとめ

本研究では継続の共有化による継続ベース Web サーバのメモリ使用量削減を提案した.

Scheme 処理系に継続の共有化機能を実装し, 継続ベースの Web アプリケーションをシミュレーションして従来の実装と提案法の実装について比較した. その結果, 提案法がメモリ使用量削減に効果があることが確認できた.

今後の課題としては, プログラムコードの静的解析により継続の共有化を自動的に行えるようにしたい.

謝辞 本研究の一部は, 科学研究費補助金 (20500028) の補助を受けている.

参考文献

[1] Queinnec, C.: The Influence of Browsers on Evaluators or, Continuations to Program Web Servers, *Proceedings of ACM SIGPLAN International Conference on Functional Programming*, pp.23–33 (2000)

[2] Clinger, W., Hartheimer,A. and Ost, E.: Implementation strategies for first-class continuations, *Journal of Higher Order and Symbolic Computation*, pp.7–45 (1999).

[3] TUTScheme: `http://www.spa.is.uec.ac.jp/~komiya/download/`

[4] Shivers, O.: The semantics of Scheme control-flow analysis, *Proceedings of the 1991 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation*, pp.190–198 (1991)