1J-03

ボランティアコンピューティングにおける ソースコードレベルのチェックポイント機能

猪原　圭一†　　黒川　陽太†　　福士　将†

山口大学大学院創成科学研究科†

1. はじめに

ボランティアコンピューティング（VC）は，インターネット上の遊休計算資源を一般ユーザから提供してもらい，大規模な分散計算環境を安価に構築する手法である．VC では，ユーザの都合により，計算ノードが計算中に離脱する問題があるため，互いに独立した複数のジョブからなる分散計算が主な処理対象とされてきた[1]．VC で並列計算を実行させようとした場合，通信相手のノードの離脱により，通信が行えずに，計算が完了しない問題を解決する必要がある．

本報告では，並列計算に対応可能な VC（並列 VC）の実現に向けて，ソースコードレベルで動作するチェックポイント手法を提案する．チェックポイントとは，ソースコード中で指定された箇所において，変数の値をファイルなどに保存し，それを復元して再開する機能を指す．提案手法では，プログラマがチェックポイント箇所と対象変数をタグにより指定するだけで，自動的にチェックポイント機能をソースコードに埋め込む．これにより，専用のソフトウェアやライブラリをインストールすることなくチェックポイント機能を実現でき，異なる環境のノードによる並列計算の引継ぎを可能にする．

2. 提案するチェックポイント機能

2.1. 概要

提案するチェックポイント機能の概要を図 1 に示す．プログラマがソースコードに以下のタグを書くことで，チェックポイント箇所と対象の変数を指定する．

#checkpoint　変数１　変数２　…

タグ付きのソースコードを変換プログラムで処理することにより，チェックポイントを実行する関数（cp_write()・cp_read()）やその関数の実行を補助する，アドレステーブルと構造体テーブルを作成する関数が埋め込まれたソースコードが自動的に生成される．

Source-code-level checkpoint functions for volunteer computing
Keiichi Inohara†, Yota Kurokawa†, and Masaru Fukushi†
†Yamaguchi University

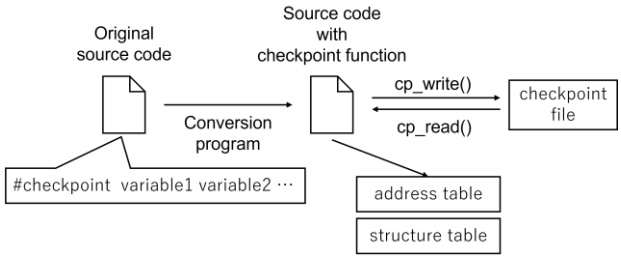

図 1　実装の概要

2.2. 変数の値を保存・復元する関数

2.2.1. cp_write()関数

cp_write()関数は，タグで指定された変数の値をチェックポイントファイルに保存する関数である．ファイルの形式を図 2 に示す．ファイルに書き込まれる値は，左から順に，変数名，要素数，変数の値である．各値は空白文字で区切られ，各変数は改行によって区別される．

保存対象の変数がポインタの場合，その値であるアドレスはノードの環境によって異なるため，アドレス値をそのまま使用することはできない．そのため，ソースコード中の全変数に対して，変数名，アドレス，要素数，型のサイズなどの情報をアドレステーブルに記録しておく．アドレステーブルで，アドレスをキーとして検索することにより，ポインタが指し示す先の変数名を取得し，図 2 の変数 c の例のように，“&変数名”の形式で，ファイルに保存する．なお，アドレステーブルに変数の情報を追加する関数として cp_addr()関数を作成しているが，詳細は紙面の都合上，割愛する．また，配列を指すポインタの場合，配列のどの要素を指しているかを明示する必要がある．このため，図 2 の変数 d の例のように，ポインタが指し示す変数名(&b)に加えて，“+配列のインデックス”の形式で，配列のインデックスも保存する．

変数が構造体の場合，構造体テーブルから変数の型を取得し，ファイルに値を書き出す．また，その構造を保つために，図 2 の変数 e のように，変数の要素ごと値を括弧で囲み保存する．なお，構造体テーブルに構造体の型とメンバー変数の情報を追加する関数として，

st_add_structure()と st_add_memner()関数を作成しているが，詳細は割愛する．

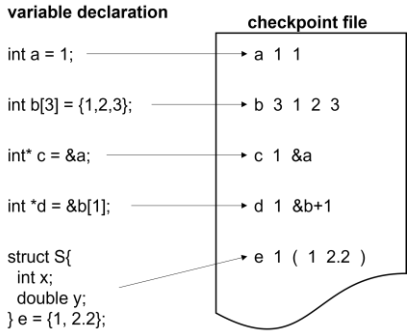

図 2　チェックポイントファイルの形式

2.2.2. cp_read()関数

cp_read()関数は，チェックポイントファイルから変数の値を復元する関数である．この関数では，まず，変数名をキーとしてファイルを検索し，該当する情報が書かれた行を特定する．次に，変数の値を文字列で読み取り，それを復元する変数の型に変換し，元の変数に値を復元する．

変数がポインタの場合，アドレステーブルを使用し，読み取った変数名をその変数のアドレスに変換する．また，配列のインデックスが保存されている場合はインデックスで指定された位置にアドレスを移動させた後で，元の変数に値を復元する．

変数が構造体の場合，構造体テーブルから変数の型を取得し，読み取った値をその型に変換する．

2.3. 指定された位置からの再開機能

プログラムを途中から再開できるようにするために，タグから次のタグまでの間の処理に対して if 文を挿入する．各タグには ID が与えられており，これを外部から入力することにより，指定されたタグよりも前の処理を if 文でスキップする．if 文の中に変数の宣言があるとコンパイル時にエラーとなるため，変数宣言を if 文の前に移動する．

3. 評価

提案手法を評価するために，cp_write()で配列の値をファイルに書き込む時間と cp_read()でファイルから読み込む時間を計測した．配列として，int 型の配列，malloc 関数で確保した配列，構造体配列の 3 種類の配列を用意し，配列サイズを 1，10，100，1,000，10,000 とした．評価環境を表 1 に示し，結果をそれぞれ図 3 と図 4 に示す．

図 3，図 4 より，3 種類の配列全てにおいて，配列のサイズが大きくなるに従い，書き込み・読み込み時間が増加することがわかる．また，配列サイズが 10,000 の場合，各配列に対する書き込み時間は，2.9ms，3.4ms，4.6ms であり，読み込み時間は，3.8ms，5.9ms，4.7ms であった．

VC で行われる計算処理の想定時間は数十分から数時間程度である．これらの結果から，提案したチェックポイント機能は，本来の計算処理に対して大きなオーバヘッドにならないことを確認できた．

表 1　評価環境

CPU	Core i7-9700
メモリ	32GB
OS	Ubuntu 20.04.3LTS

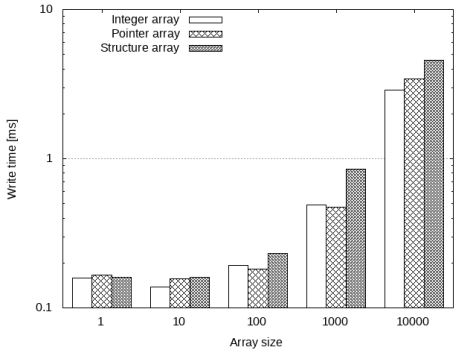

図 3　書き込み時間

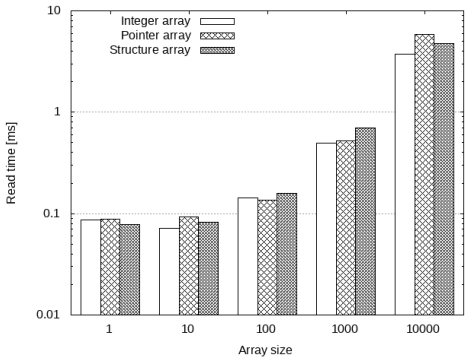

図 4　読み込み時間

4. おわりに

本報告では，並列 VC の実現に向けて必要になるソースコードレベルのチェックポイント機能を提案，実装し，評価により，チェックポイントの実行時間が本来の計算処理に対して大きなオーバヘッドにならないことを確認した．

今後は，実際の並列分散計算を対象として，チェックポイントが全体の計算時間に与える影響を評価する予定である．

参考文献

[1] Folding@home,　https://foldingathome.org