1K-7

NFS 上のファイルアクセスを考慮したシェルスクリプトの並列実行手法の提案

†前田　直人　‡水谷　泰治
†大阪工業大学　大学院情報科学研究科
‡大阪工業大学　情報科学部情報システム学科

1 はじめに

シェルスクリプトは UNIX 系の OS で広く利用されている標準的なスクリプト言語である．近年，シェルスクリプトで大規模な処理が行われている．例えば文献[2]では，大学のプログラミング演習において学生の提出したプログラムの採点をシェルスクリプトで行っている．このシェルスクリプトを高速に採点できれば演習中に迅速に適切な指導ができるため，シェルスクリプトの高速化が望まれている．そこで我々はシェルスクリプトを PC クラスタ上で手軽に並列化できるツールの開発を行った[1]．このツールによりシェルスクリプトの繰り返し文を並列に処理することにより作業の高速化を狙う．

しかしこのツールは，逐次処理と比べて速度向上率が理想値の半分にとどまり高いとはいえない．また、スクリプト内での NFS へのアクセス頻度の違いにより，速度向上に差が出てしまう．そこで本研究では，NFS 上のファイル操作による遅延を隠蔽することを目的に，ワーカ上のタスクの多重化及びキャッシュという手法を提案する．

2 シェルスクリプト並列化ツールの検討

2.1 既存ツールの概要

本ツールは，少ない手間によりシェルスクリプトの並列処理を行えることを目的としている．for 文の前後をユーザが特殊なコメントで囲み，並列処理用のスクリプト群を自動生成する．そのスクリプトを実行することで，for 文の繰り返し 1 回を 1 つのタスクとしてマスタ・ワーカ法(MW 法)を用いて並列処理を行う．詳細は文献[1]を参考されたい．本ツールの使用環境として LinuxOS の PC クラスタを使用した．PC クラスタの各ノードでは，NIS によってアカウントを一元管理し，NFS によってホームディレクトリを共有している．

次に性能について述べる．文献[1]より，NFS へのアクセス頻度が多いほど性能が低くなることが分かっている．そこで，2 種類のスクリプトを用意し性能を比較した．1 つ目は，I/O が主であるプログラムとして，プログラミング演習にて学生の作成した課題プログラムを採点するプログラム（test.sh)である．文献[2]を参考に模倣して作成した．2 つ目は，CPU が主であるプログラムとして C 言語で作成した行列積計算を繰り返し実行するプログラム(dummy.sh)である．図 1 は test.sh の概要である．図 2 のように格納されている各学生のプログラムを参照し，キーワードの検索，コンパイル，出力の検査を行い，課題の提出物として不都合があればエラーメッセージをログに記録する．

```
#!/bin/sh
#parstart 並列化への特殊コメント
for StudentNo in 学生番号のリスト;do
   for File in 課題ファイルのリスト;do
      while read Keyword #キーワード読込;do
         #ファイルにキーワードがないとき キーワード >> ログ
      #エラーテキストがないならコンパイル
      while read Input #実行ファイルに入力するデータ読込;do
         #実行.エラーのとき 入力データ >> ログ;break
         #正解と違うとき 正解 >> ログ;break
#parend 並列化への特殊コメント
(紙面の都合上 do に対する done コマンドは省略)
```

図 1　採点用プログラムの概要(test.sh)

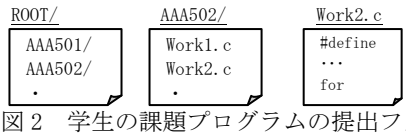

図 2　学生の課題プログラムの提出フォルダ

本ツールでtestとdummyの2種類のスクリプトの速度向上率をワーカPC10台で検証した結果，前者が6.5倍に対して後者が9.9倍となった．この結果よりNFSサーバへのアクセス頻度が少ないほど理想値に近い値が出ることが分かった(図3)．そこで本研究ではNFSサーバへのアクセス頻度を減らすために，多重化とキャッシュを提案する．

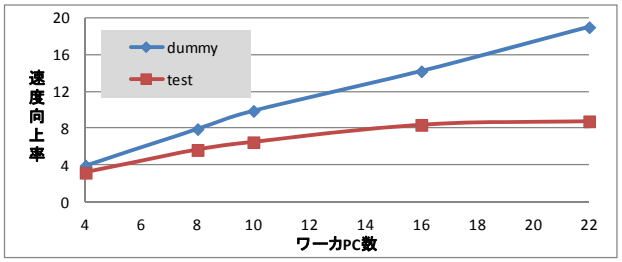

図3　改良前の性能評価

2.2 多重化

多重化とは，1台のワーカPCで複数のワーカプロセスを起動し同時に複数のタスクを実行させるようにしたものである(図4)．これによりNFSサーバとの通信によるワーカのオーバヘッドを隠蔽する．

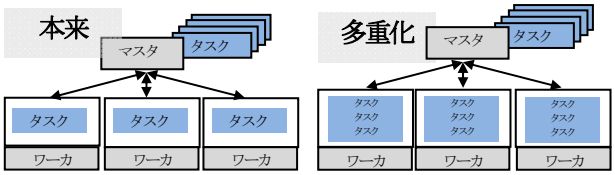

図4　多重化の概要

2.3 キャッシュ

キャッシュとは，スクリプト内でアクセスの多いファイルを各ワーカのローカルディスクに一時保存する作業を指す．並列実行の際，キャッシュしたファイルを参照することでNFS上のファイルとのアクセスを減らす．図5に示すようにキャッシュの適応には，スクリプト中にキャッシュ用特殊コメントとファイル名を記入する(A)．これによりNFSサーバから各ワーカへ一時保存され(B)，生成されたスクリプト内のパスがローカルディスクに変更される(C)．新規に生成させるファイルには(B)の処理はなく，ローカルディスクに新規生成する．ファイルがキャッシュに適しているかはユーザが判断する．

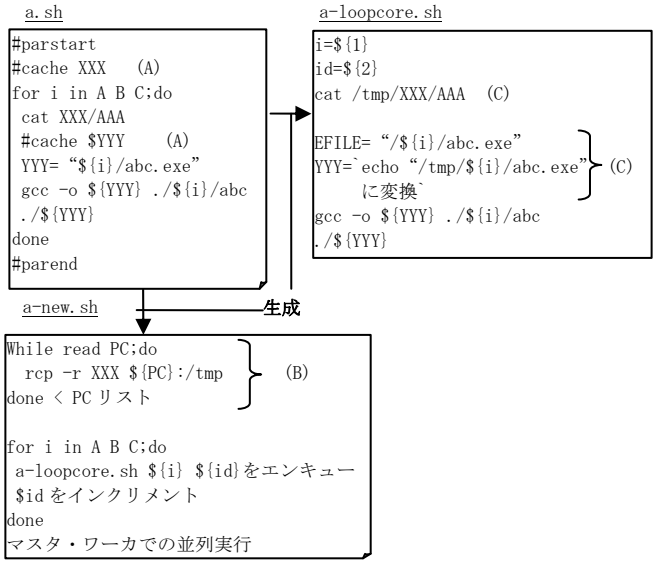

図5　キャッシュを考慮した変換

3 性能評価

改良前のツールと今回の改良後のツールを段階的に比較し，どれだけ速度が向上したのかを調査する．また，逐次プログラムにキャッシュを適応した場合との比較を行い，並列処理での多重化，キャッシュの有効性を調査する．

3.1 結果と考察

2節に述べた方法により，逐次実行した場合と同等の実行結果が得られたことを確認した．以降は性能について述べる．

図6にワーカPC10台で各々の改良案を適応させたtest.shの性能評価を示す．なお，多重化とキャッシュの組み合わせを凡例①～④の通りとし，縦軸は凡例①に対する速度向上率，横軸は凡例①～④とする．速度向上率は，既存ツールと比べそれぞれ1.3倍，1.9倍，2.4倍であり，全ての場合で改善した．また，図7に逐次でキャッシュを適応した際の適応前との速度向上率の比較を示す．速度向上率は，逐次は1.4倍となり本ツールでキャッシュを適応した際の1.9倍の方がよい結果となった．この結果より，キャッシュによる元々の性能向上以外にも，本ツール特有の性能向上に関する要因があると考えられる．その要因として考えられるのは，NFSサーバとの通信によるオーバヘッドの隠蔽が影響だと推定する．

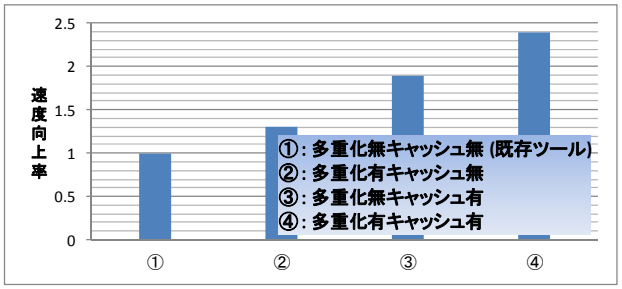

図6 改良後との性能評価

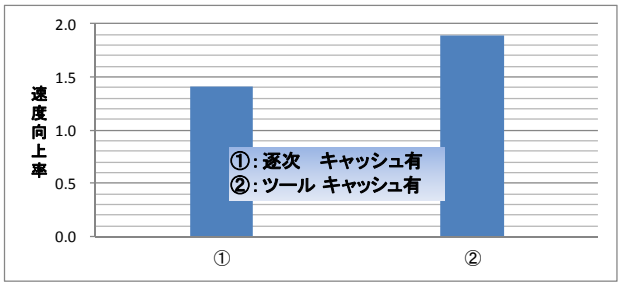

図7　キャッシュにおける逐次との性能評価

4 まとめ

本研究では，文献[1]の手法に対して多重化とキャッシュという手法を提案した．また，実験の結果，実アプリケーションにおいてワーカPC10台を用いて既存ツールと比べ2.4倍の速度向上率を得ることがわかり，一定の有用性はあると我々は考える．しかし，キャッシュするファイルの選択はユーザに任せているので少ない手間で並列処理を行えるとは言えない．今後は，キャッシュするファイルを自動的に判別させることが課題である．

謝辞　本研究の一部は，科学研究費補助金基盤研究(B)(21300011)の補助による．

参考文献

[1]菅圭佑他“繰り返しに着目したシェルスクリプトの半他自動並列化手法の提案”．第15回 電子情報通信学会関西支部 学生会研究発表会，p.79，(2010)
[2] 内藤広志他“プログラミング演習の進捗モニタリングシステムの評価”．第7回情報科学技術フォーラム講演論文集(FIT2008)，pp.319-320，(2008)