関数の処理結果を永続的にキャッシュするシステムの設計と実装

情報科学科　石川 雄基　　指導教員：山村 毅

1　はじめに

プログラミングを用いる情報科学などの分野の研究においては，入力したデータから必要とする結果を求めるまでに，複数回にわたって処理を行わなければならない場合がほとんどである．しかし「ある一部の処理のパラメータ調整」のような試行錯誤の過程においては，その他の大部分の処理では「同じ入力データに対して同じ処理を行い同じ結果を返す」という冗長な処理を行っていることがある．

本研究では，研究開発における試行錯誤の過程の処理時間の短縮や煩雑なデータファイル管理を容易にすることを目的として，関数のキャッシュシステムを提案する．

2　提案手法

2.1　システム概要

本研究のモジュール (以下，本モジュール) の概要を図 1 に示す．

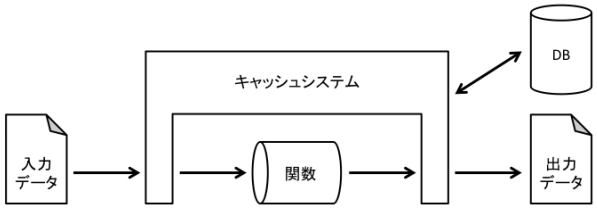

図 1　キャッシュシステムの概要

まず，キャッシュしたい関数の関数呼び出しをフック[*1]する．次に，その関数呼び出しに対応する処理結果がキャッシュされているかどうかを確認する．キャッシュされていた場合，関数は実行せずにキャッシュデータベースから処理結果を読み込んで返す．キャッシュされていなかった場合，関数を実行した後で処理結果をキャッシュデータベースに保存する．

2.2　スキーマ設計

本モジュールのキャッシュデータベースには，汎用的な RDBMS である SQLite を利用した．そのスキーマを表 1 に示す．

表 1　キャシュテーブルのスキーマ

カラム名	データ内容
method	関数名 / メソッド名
checksum	引数のチェックサム
result	処理結果

保存する情報は「キャッシュデータを特定するために必要な情報」と「キャッシュデータ」の 2 種類である．また前者の情報の中でも，引数の情報はそのまま保存するとデータサイズが大きくなってしまう可能性があるため、引数の情報は MD5 チェックサムの値を計算して保存する．

[*1] 関数呼び出しを横取りすること．処理の前後に任意の処理を追加したり，引数や返り値を操作したりすることができる．

3　導入方法

本モジュールは Cache::Method という Perl モジュールとして実装した。例として，本モジュールを利用して関数 foo の処理をキャッシュする Perl スクリプトをソースコード 1 に示す．

ソースコード 1　関数 foo をキャッシュする例

```
1  use Cache::Method;
2
3  my $c = Cache::Method->new( dbfile => 'cache.db' );
4  $c->set('foo');
5
6  sub foo { return 'foo!'; }
7
8  print foo(); # Call foo()
9  print foo(); # Cached result
```

1 行目でモジュールの読み込み，3 行目でデータベースファイルの指定，4 行目でキャッシュする関数の設定を行っている．基本的にはこれだけの記述で本研究で提案するキャッシュシステムを利用することができる．

4　評価実験

フィボナッチ数列の計算を，本モジュールを使用した場合と使用しなかった場合とで処理時間を比較した．結果を図 2 に示す．

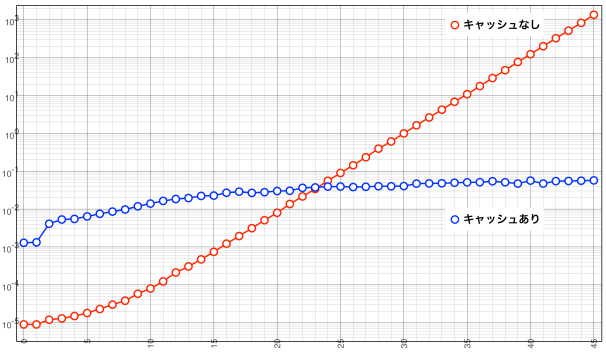

図 2　フィボナッチ数列の処理時間

本モジュールを使用しなかった場合では項数が増えると処理時間が爆発的に増加するのに対し，本モジュールを使用した場合では項数が増えても処理時間がほとんど変わらないことがわかる．

5　考察

評価実験の結果では，項数が小さいときはキャッシュしないほうが処理時間が短かった。これはモジュールの読み込み時間やデータベースアクセスの時間が原因だと思われるが，時間がかかる処理に対してはこれらのオーバーヘッドはほとんど無視できるレベルである．したがって，複数回呼び出される時間がかかる処理に対しては本モジュールは極めて有効であると言える．