情報ネットワーク学演習 II
第2回レポート課題

所属：大阪大学 大学院情報科学研究科 情報ネットワーク専攻
提出者: 33E16019 満越貴志
電子メールアドレス: t-mangoe@ist.osaka-u.ac.jp

提出年月日：平成 28 年 10 月 12 日

課題内容

Ruby のプロファイラを用いて、Cbench のボトルネックを解析する。

Cbench のボトルネックの解析

Ruby のプロファイラの 1 つである profile を用いて、ボトルネックの解析を行った。cbench リポジトリの lib 配下の cbench.rb ファイルに

```
require 'profile'
```

の記述を追加し、Cbench を実行した。実行結果は以下のようになった。

```
199.87   251.42    251.42         2 125710.00 125710.00  IO.select
 99.99   377.20    125.78       142    885.77    885.77  Kernel#sleep
 99.94   502.91    125.71         2 62855.00 62855.00  Thread#join
 99.94   628.62    125.71         2 62855.00 62855.00  TCPServer#accept
  3.84   633.45      4.83    115947     0.04      0.06  Kernel#dup
...(omitted)
```

この結果より、

- IO.select
  与えられた入出力待ちの IO オブジェクトの中から、準備ができたものを配列の配列として返す。
- Kernel#sleep
  プログラムを停止するメソッド。他スレッドから起こされると、プログラムを再開する。
- Thread#join
  生成したスレッドの終了を待機するメソッド。
- TCPServer#accept
  クライアントからの接続要求を受け付け、接続した TCPSocket のインスタンスを返すメソッド。

の４つのメソッドが、Cbench の実行時に多くの時間を消費していることが分かった。この４つのメソッドが、ボトルネックになっていると考える。

ここで、Kernel#sleep メソッドと Thread#join メソッドは、どちらもスレッドに関連するメソッドである。また、IO.select メソッドも、入出力待ちのスレッドを管理するためのものではないかと思われる。スレッドの待機や終了が頻発していることが、プログラムの実行の遅延につながっていると考える。

参考にしたサイト

- Ruby 2.3.0 リファレンスマニュアル
  https://docs.ruby-lang.org/ja/latest/doc/index.html