情報ネットワーク学演習　レポート課題３

氏名: 佐竹　幸大
学籍番号: 33E16010

課題内容 (Cbench のボトルネック調査)

Ruby のプロファイラで Cbench のボトルネックを解析しよう。
以下に挙げた Ruby のプロファイラのどれかを使い、Cbench や Trema のボトルネック部分を発見し遅い理由を解説してください。

解答内容

ruby-prof を使い、ボトルネックの調査を行った。
実行の手順は以下の通りである。

・インストール

gem install ruby-prof

・プロファイルの結果出力処理

ruby-prof ./bin/trema run ./lib/cbench.rb > cbench-profile.txt

・cbench の実行

./bin/cbench --port 6653 --switches 1 --loops 10 --ms-per-test 10000 --delay 1000 --throughput

cbench の実行終了後、プロファイル結果が cbench-profile.txt に出力される。

ここで、cbench.rb 実行時のプロファイル結果のうち、実行時間が長い上位 10 件の情報を以下に示す。

Thread ID: 17684720
Fiber ID: 17684520
Total: 169.690853

%self total self wait child calls name
2.86 32.822 4.846 0.116 27.860 113441 BinData::Struct#instantiate_obj_at
2.68 4.551 4.551 0.000 0.000 136912 Symbol#to_s
2.65 4.493 4.493 0.000 0.000 127525 BasicObject#!
2.23 19.595 3.786 0.000 15.809 168853 *BinData::BasePrimitive#snapshot
2.14 16.494 3.627 0.000 12.867 163604 *BinData::BasePrimitive#_value
1.98 27.616 3.360 0.163 24.093 70570 BinData::Struct#find_obj_for_name
1.82 4.554 3.092 0.000 1.461 99144 Kernel#define_singleton_method
1.75 7.427 2.968 0.000 4.459 63527 Kernel#clone
1.72 2.917 2.917 0.000 0.000 70578 String#sub
1.71 3.112 2.895 0.000 0.217 101918 BinData::Base#get_parameter

このうち、実行回数を表す calls の値が最大となるのは、*BinData::BasePrimitive#snapshot である。次いで*BinData::BasePrimitive#_value の呼び出し回数が多い。このことから、BinData に多くの時間がかかっており、これがボトルネックになっていることが推測できる。次に、BinData が、コントローラにおける Cbench の処理のうち、どの箇所で生成されているのかを考えた。その結果、BinData を継承した Format が、FlowMod うちでは生成されていることがわかった。FlowMod の生成は、send_flow_mod_add で生成されていることから、send_flow_mod_add がボトルネックになっているのではないかと考えられる。

次に、cbench を高速化した fast_cbench.rb を実行した結果が以下である。

Thread ID: 13840260
Fiber ID: 13840120
Total: 115.816474

%self total self wait child calls name
3.31 3.829 3.829 0.000 0.000 170741 Symbol#to_s
3.27 26.968 3.789 0.093 23.085 127733 BinData::Struct#instantiate_obj_at
3.24 11.024 3.748 0.000 7.275 162921 *BinData::BasePrimitive#snapshot
3.01 3.487 3.487 0.000 0.000 154618 BasicObject#!
2.94 4.963 3.401 0.114 1.448 131262 Kernel#define_singleton_method
2.16 14.468 2.501 0.000 11.967 82564 BinData::Struct#find_obj_for_name
2.09 8.152 2.418 0.000 5.734 82564 BinData::Struct#base_field_name
2.08 4.157 2.405 0.000 1.752 112066 *BinData::BasePrimitive#_value
1.96 2.473 2.274 0.000 0.199 134679 BinData::Base#get_parameter
1.93 77.772 2.239 0.000 75.533 55305 *Array#each

実行回数を表す calls の値が前述の cbench.rb の結果と比較の 163604 と比較すると、112066 に減少しており、実行時間の削減ができているとわかる。

参考文献

[1] 情報ネットワーク学演習 2 事前準備 https://github.com/handai-trema/syllabus#事前準備
[2] Trema で OpenFlow プログラミング http://yasuhito.github.io/trema-book/#cbench