ソフトウェア・クラウド開発プロジェクト実践III 6/3課題

村上恵太郎, 48226624, 創造情報学専攻

MacOS上にUbuntu18.04のDocker環境を構築し，Pythonによるベンチマークプログラムを走らせた．プログラムはこちらである(https://github.com/ketaro-m/utokyo-ci-sdp-docker/blob/main/bench.py)．以下のテストを行って実行時間を計測してファイルに書き出した．

1. 空ループを1000万回
2. 関数を100万回呼出し
3. print を10万回
4. ジュリア集合の計算
5. スレッド生成10万回
6. プロセス生成10万回

結果は以下のようになった．1-4の処理では，パフォーマンスに大きな差異は見られなかった．
スレッド生成やプロセス生成といったシステムコールを用いるものではネイティブOSで実行した方がパフォーマンスが良いという結果が出た．Pythonのスレッドはプロセスでシミュレートしたものではなく，Unix/Linux系ではPOSIXスレッドを，WindowsではWindowsスレッドという，完全にOSによって管理されたスレッドを用いているので，このような差が出たものと考えられる．printはOSのシステムコールを呼んでいるが，Docker上の方が僅かにパフォーマンスが上がるという結果が出た．

—----------

MacOS

—----------

Python:
 - Python: 3.8.6 (CPython)
 - Build: ('default', 'Feb 4 2021 00:50:13')
 - Compiler: Clang 12.0.0 (clang-1200.0.32.29)
PC Info:
 - OS: macOS-11.5.2-arm64-arm-64bit
 - Processor: arm
   - Core: 8/8
 - Memory: 8GB (8,589,934,592 Byte)

loop_test: 103.6958 ms
function_call_test: 58.03099999999999 ms
print_test: 350.967 ms
calc_julia_set: 1681.8700999999996 ms
Thread: 582.4029 ms
Process: 8478.3633 ms

—--
Ubuntu 18.04 (Docker on Mac)
—--
Python:
 - Python: 3.8.6 (CPython)
 - Build: ('default', 'Jul 3 2022 16:35:46')
 - Compiler: GCC 7.5.0
PC Info:
 - OS: Linux-5.10.104-linuxkit-aarch64-with-glibc2.27
 - Processor: aarch64
   - Core: 4/4
 - Memory: 4GB (4,124,368,896 Byte)

loop_test: 110.3108 ms
function_call_test: 53.67659999999999 ms
print_test: 303.9207 ms
calc_julia_set: 1719.0277999999998 ms
Thread: 884.8479 ms
Process: 9299.239800000001 ms