システムプロプログラミング特論 課題 2-2

～演算コストの計測～

1. はじめに

システムプログラミング特論課題 2-2 の提出物である。課題 2-2 の内容と提出物の要件は下記の通り。

【課題内容】
講義資料の演算コスト表を参考にして、あらかじめアルゴリズム、データ構造、計算量、プログラミング言語、言語処理系等の何らかの比較を考慮し，実際に複数の演算のコストを計測し、性能の相違・特徴等を考察してください。

【提出物】

- テストコード（抜粋）
- コスト表
- 考察及び工夫した箇所

今回選択したてーまおよび概要は下記の通り。

【テーマと概要】

- テーマ：Python と Java の実行速度を比較
- 概要：Python と Java で同様の処理を行うプログラムを用意し、その実行速度を比較することで両言語の特徴を考察する。そのため、繰り返し処理による数値の階乗と、処理による数値の加算をテストする。

2. 比較したプログラムのコード

Python および JAVA で下記のようなコードを用意した(サンプルは Python)

2.1. サンプル１：階乗を繰り返す

```
x = 1
count = 1

while (count < 25):
        count = count + 1
        x = (x + 1) * x

print("計算結果を出力")
print(x)

print("繰り返し回数を出力")
print(cont)
```

2.2. サンプル 2：1 億回加算する

```
x = 1
count = 1

while (x < 1000000000):
        x = x + 1
        count = count ; 1

print("計算結果を出力")
print(x)

print("繰り返し回数を出力")
print(count)
```

3. コスト表

テストの結果を下記の表にまとめた。テスト方法/テスト環境などについて詳細は別に記載する。

項番	言語	コード名	処理の内容	時間
1	Python	sample_001_cal.py	階乗を繰り返す (25 回)	3 分 14.968 秒
2	JAVA	sample_001_cal.java	階乗を繰り返す (25 回)	0.104 秒
3	Python	sample_002_cal.py	加算を繰り返す (1 億回)	1 分 58.192 秒
4	JAVA	sample_002_cal.java	加算を繰り返す (1 億回)	0.484 秒

4. テスト環境

4.1. ハードウェア

- 環境構築先：Google Cloud Platform
- マシンタイプ：n1-standard-1 (Google Cloud Pltform での名称)
- CPU：vCPU x 1
- メモリ：3.75 GB
- リージョン：us-east1-b (Google Cloud Platform 上では「ゾーン」と記載)

4.2. OS

- OS：Devian
- バージョン：4.9.0-6-amd64

4.3. ソフトウェア

- Python：バージョン 3.5.3
- JAVA
  - JRE(Java Runtime Environment)：build 9

- JDK(Java Development Tool Kit)：build 9

5. テスト方法

コストをテストする際には bash にて time コマンドを入力しプログラムを実行、その実行結果の中から CPU 時間を抜き出した。

【コマンド例】
time python3 sample_001_cal.py

【出力例】
Real　　0m59.037s (実時間)
User　　0m58.997s (ユーザ CPU 時間)
sys　　0m0.029s (システム CPU 時間)

今回はプログラミング言語ごとの処理速度を計測するため、プログラム自体が CPU を使用している時間である「ユーザ CPU 時間」を使用した。

6. 考察

結果の数値だけを参照すると Java の方が処理速度が速いように見えるが、乗算の出力結果は全く異なる。
JAVA では int 型/Long 型/double 型などの範囲内でのみ処理され出力される。
Python3 では整数型が int 型に統一され桁数の上限が廃止されている。そのため項番 1/項番 2 においてはすべての桁が出力された。
データ型の仕様上、JAVA は Python に比べて直接の計算処理には不向きと考えられる。
加算のコストについては、J 出力結果が同じであるにも関わらず JAVA のほうが上回ったこれはコンパイルされている分だけ Python より高速に処理できたものと考えられる。

7. 工夫したところ

演算コストの差を明らかにするため、可能な限り計算量が多くなるようにした。
今回は「繰り返し処理による階乗」と「繰り返し処理による加算」の二種類のサンプルを用意したが、それぞれ「25 回繰り返す」「1 億回繰り返す」というように設定している。
また、処理が正しく行われていることを確認するために繰り返し回数を計測し出力できるようにした。
ハード面について、実機や実機内に設置した仮想環境を使った場合は端末の予期せぬ動作で結果に差が出る可能性がある。対策として Google Cloud Platform 上に最小構成の仮想マシンを用意しそこでテストをすることで誤差を排除するようにした。
また、テスト環境や使用したコマンドを提出物に明記することで容易に追試が行えるようにしている。