第４回　コンピュータ科学基礎論　資料　　2017-10-13 山内

今回は、並列プログラミングのモデルについて、議論したい。
プログラミング言語は、それが想定するモデルが、プログラムの形や実行の形式を規定する側面がある。 つまり、プログラマは言語が想定するモデルの範囲内でプログラムを作るように、束縛される。 並列処理を提供する言語のあり方について考えてみたい。

(1)　並列記述のできるプログラミング言語について、その想定する並列モデルを整理してみよう。 どのようなものがあるか、具体的にどのような書き方を提供しているか （pthread、OpenMP、MPI、論理型言語、（関数型言語）、他にあれば何でも）

(2) （脱線かもしれないが） プロセス間の同期について、fork-join （バリヤも含める） のような取り方　vs　セマフォのような取り方　の比較をしてみよう。

(3) プログラミング言語の提供する並列モデルとハードウェアの関係は、どう考えるのが良いだろうか。
たとえば OpenMP vs MPI と、共有メモリ vs 分散メモリ とは、どの程度「密」に対応しているだろうか。

他方、論理型言語のような高級な（＝ハードウェアから見ると抽象度の高い）言語は実行時のハードウェアの構成を想定しない（少なくとも建前上は）と言えるだろう。 ユーザはハードウェアとの対応をどの程度意識すべきなのだろうか？　意識せずにプログラムできるのが良いか、意識して効率のよい使い方を考えるのが良いか？　将来どちらへ行くべきだろうか？

（論理型の言語の例:　Prolog）

```
like(taro, coffee).
?- like(taro, Y).
Y = coffee
```

```
like(taro, coffee).
?- like(X, coffee).
X = taro
```

```
like(taro, coffee).
like(hanako, coffee).
?- like(X, coffee).
X = taro
X = hanako
```

```
like(taro, coffee).
like(hanako, coffee).
like(taro, cocoa).
?- like(X, Y).
X = taro
Y = coffee
X = hanako
Y = coffee
X = taro
Y = cocoa
```

(4)　手続き型言語でも、データに対する処理の多変数（ベクトル）に対する同一計算が使える設定がある。
（Python の numpy パッケージを使った時の例）

```
import numpy as np
x = np.array([0, 1, 2, 3, 4])
y = np.array([6, 7, 8, 9, 10])
for i in range(5):
    print(x[i]+y[i])
```

⇨

```
import numpy as np
x = np.array([0, 1, 2, 3, 4])
y = np.array([6, 7, 8, 9, 10])
print(x + y)     ← i を指定していない
```

ループを書かないことで並列処理の可能性があるという見方もできると思うのだが、どうだろうか？　考えてみて欲しい。
同様なことは、既存言語でも関数 ArrayAdd を定義したり、オブジェクト指向環境で演算子＋をオーバーライドしたりすることで実現でき、実際に行列計算ライブラリなどで行われている。