第3章 スレッドのスリープや待ち合わせ

並列処理を自分で細かくプログラムする方法

佐藤 聖

● 複数のコアに処理を分散する

例えば，ラズベリー・パイなどではメイン・プロセスとサブ・プロセスで異なるCPUコアで処理したい場合があります．これまでも本誌の中でラズベリー・パイで機械学習モデルを利用したり，GPIOでLEDやモータ・ドライバを制御したりする際に並行処理や並列処理を取り入れてきました．PCよりもCPU処理性能が低いので1つのCPUコアで集中的に処理するより，複数のCPUコアに分散して処理した方が，高速に動作することがあります．

扱いやすいモジュールで並列処理

ここではシンプルなソースコードが書けるPythonモジュールのthreading，multiprocessing，Joblibを使って並列処理する例を紹介します．

Python標準にもconcurrent.futures関数があり，並行処理のThreadPoolExecutor，並列処理のProcessPoolExecutorといった機能があるため，並行処理も並列処理も実装できます．多くの機能が提供されていて便利ですが，その代わりソースコードの記述はやや複雑になります．これらと比較するとthreading（並行処理）やmultiprocessing（並列処理）の方が扱いやすいです．

いずれもメイン・プロセスとして，全体の処理を制御し，サブプロセスでタスクを実行するケースでよく使われます．

● その1：スレッド・ベースの並列処理

Python標準モジュールで，スレッド・ベースの並列処理が行えます．実際はPythonのスレッドの安全性を確保する機能が働くため，並行処理になります（第1章参照）．

高水準のスレッド・インターフェースを低水準の_threadモジュールの上に構築しています．共有されたメモリからデータをスレッド群に分配して命令を同時実行します．並列処理でよく利用される高水準の関数が用意されており，短いソースコードでも簡単にスレッドをコントロールできます．

● その2：プロセス・ベースの並列処理

Python標準のモジュールで，プロセス・ベースの並列処理が行えます．スレッドの代わりにサブ・プロセスを使用します．複数のデータをサブ・プロセス群に分配し，各メモリ空間に格納されたデータを対象にして関数で並列に処理します．

● その3：科学技術計算などの大規模演算向け

Pythonで軽量のパイプライン処理や並列処理が行えます．高速に大規模データセットを処理する科学技術計算に向いています．並列処理に関する便利な機能があり，用途に応じた高速化が可能です．同時に複数の演算処理に適した並列化，構文解析にも使われます．

計算結果を一時的に保持し，重複する引数による関数の再計算を防ぐ手法のメモ化や，大容量の計算結果を圧縮することで1つのまとまりとして扱い，保存に適した直列化（シリアライズ）を行う機能があります．

処理の流れ

いずれのプログラムもメイン・プロセスまたはメイン・スレッドから3つのサブプロセスまたはサブスレッドを生成して関数func1，func2，func3を実行します．

定義した関数内のソースコードをGoogle Colaboratory上で書き換えて，読者が練習できるようにしました．

関数func1，func2，func3の中でtime.sleep関数とprint関数を実行します．関数が実行されると，それぞれprint関数の引数に与えた文字列を標準出力に出力します．

一般的なプログラム開発でthreadingとmultiprocessingを使う場面はもっと複雑な処理となる場合がほとんどですが，基本的な使い方はここで紹介するプログラムと一緒です．参考にしていただき，