第5章 x86/Arm/NVIDIA GPUで高速化

ご購入はこちら

SIMD/メモリ/AI機能…CPUに用意された拡張命令

本橋 弘臣

5-1 1つの命令で複数のデータを同時に処理するSIMD（Single Instruction Multiple Data）命令

Q1 最適化

Q ソフトウェア最適化にはどのような手法がある？
A アルゴリズム/データ構造の改善をはじめとするさまざまな手法がある

ソフトウェアの最適化手法には，**表1**に示すような方法があります．この中で一番重要なのは，①アルゴリズム/データ構造の改善です．②～⑥のある意味で小手先の改善を実施したとして，そもそも採用したアルゴリズムが不適切なものだったとすると，最適化のために実施した努力/労力は全て水の泡になってしま

表1 ソフトウェアの最適化手法

番号	最適化手法	適用対象例	対処方法	効 果
①	アルゴリズム/データ構造の改善	• 大量データに対する検索処理をリニア・サーチで実装したら遅かった • 配列に対するデータの挿入/削除処理が遅い • 巨大なデータをソートするのが遅い	• 2分検索に変更する • ハッシュ・テーブルの活用 • 配列をリスト構造に変更する • 巨大なデータそのものを入れ替える代わりに，データを指すポインタのみをソートする	×2～10000
②	ループ内での無駄な処理の削除	• 計算結果が変わらない計算をループ内で毎回行っている	• ループごとに計算し直す必要がない計算は，ループの外側で実施する	×2
③	CPUが苦手な演算/処理の排除	• 割り算は遅い • 掛け算もそこそこ遅い • 実はCPUは分岐命令が非常に苦手．特に実行回数の多いループ内で分岐処理を行うと，致命的と言えるほど遅くなる • 画像処理プログラムの最内周ループ内で動作モードを判定して処理内容を切り替えるようにしたところ，画像処理速度が非常に低下した	• 除算は極力行わない．どうしても除算が必要な場合は，逆数の掛け算に変更する • 掛け算を加算の繰り返しに置き換える • 実行速度が重要なループ処理の内部では分岐命令は使用しない．何とかして他の方法でコードを記述する • 共通関数で複数の動作モードに対応させる代わりに，動作モードごとに関数を用意する	×2～10
④	メモリ・アクセスの最適化	• ランダム，あるいは飛び飛びのアドレスのデータにアクセスすると，キャッシュのヒット率が低下して，プログラムの実行が遅くなる • 場合によってはキャッシュ・メモリを利用することで，かえってメモリ・アクセス性能が低下することがある	• できるだけ連続したアドレスのデータにアクセスするように，プログラムの作りやデータの処理順序/配置を工夫する • キャッシュ・メモリをバイパスしてメモリにアクセスすることが可能な特殊なメモリ・アクセス命令を利用する	×2～×16
⑤	ベクトル演算（SIMD）命令の活用	• 巨大な画像データに対する画像処理を行うために1画素ずつ順繰りに処理するプログラムを作ったところ，非常に遅かった	• 32画素分の画像データに対する演算処理を一度にまとめて実行するようにプログラムを書き変える（=SIMD命令の活用）	×4～32
⑥	マルチスレッド処理の採用	• プログラムがシングル・スレッド処理で実装されているため，せっかくたくさんのCPUコアがあるのに，1コアしか利用されない	• プログラムが片付けるべき仕事を複数のスレッドで分担して，同時並行的に処理するようにプログラムを書き換える	×2～8
⑦	各種専用演算命令の活用	• ディープ・ラーニング技術による AI 推論処理や，生成AI処理をCPU上でのソフトウェア処理で実装したところ，満足な性能が出なかった • AESの暗号化/復号処理をCPUで行うと，処理性能が低い	• AI推論処理用のアクセラレータであるNPUの活用 • 行列演算専用のアクセラレータであるインテルAMX命令[注]の活用 • AES演算専用命令の活用	×4～1000

注：AMX：Advanced Matrix eXtension