第1章 AI処理で多用する行列演算でも使える

プリフェッチ命令で処理を高速化する

松岡 洋

キャッシュを使ってメモリの速度を補う

● CPUとメイン・メモリの速度差は大きい

製造プロセスの進歩や，拡張命令の導入などによって，演算処理の高速化が行われてきました．しかし，それらによる高速化も頭打ちになってきています．特に，画像AIやLLM（Large Language Models）のように大きなデータを扱う場合は，プロセッサとメモリとのデータ転送がボトルネックになるため，HBM（High Bandwidth Memory）のような広帯域のメモリを使う方法がありますが，極めて高コストになるため，適応できるアプリケーションは限られています．

CPUやGPUは数GHzのクロックで動作しており，内部ではns単位で処理が進みます．しかし，どのような計算をするにしても，外部のメモリからデータを読み込む必要があり，データが届くまで約300クロック（3GHzクロックの場合は100ns）程度の時間がかかります．そのため，メモリからのデータ転送が処理全体のボトルネックとなっています．

● CPUが持つ速度差を補う仕組み

メモリにアクセスしデータが届くまでには100nsの待ち時間が発生し処理が停滞します（ストールと呼ぶ）．処理に条件分岐がある場合，次に実行する処理を事前に確定できないので，分岐後のアドレスからプログラムを読み込むためにストールが生じます．これを回避するためCPUには次のような機能が組み込まれています．

- 階層化されたキャッシュ
- ハードウェア・プリフェッチ
- 分岐予測

インテルおよびAMDのCPUはL1，L2，L3という3層構造のキャッシュ・メモリ（以降，キャッシュ）を持っています（**図1**）．高速かつ小規模なL1キャッシュではデータ用と命令用は別々なキャッシュになっており，それぞれL1d，L1Iと呼ばれています．

L2キャッシュはCPUのコアごとに組み込まれており，それぞれが数Mバイトの容量を持ちます．

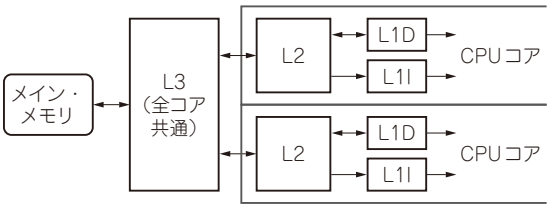

図1 CPUキャッシュは階層構造になっている

L3キャッシュは数十Mバイトの容量を持ち，全てのCPUコアで共有されています．L1キャッシュやL2キャッシュに比べると動作が遅く，アクセスに数十nsかかります．

CPUがメモリからデータを読み出す場合，L1キャッシュにデータがあれば，短時間で読み出せます．L1キャッシュにデータがない場合（キャッシュ・ミス），自動でL2キャッシュからL1dキャッシュに転送されます．L2キャッシュとL3キャッシュの関係も同様です．

自動でストールを防ぐ仕組み

● メモリ上に連続配置すればストールしない

メイン・メモリ上に連続して配置されている命令やデータに対して連続してアクセスする場合は，ハードウェア・プリフェッチャによって命令実行前に先読みが行われます．これによって命令そのものや命令実行に必要なデータが，メモリからキャッシュへあらかじめ読み込まれます．

連続したデータにアクセスする場合は非常に高速に処理ができる一方で，メモリ上に連続して配置されていないデータにアクセスする場合には，先読みによってキャッシュにデータを用意しておくことができないので，データの読み出しに時間がかかります．その結果，直ちに演算を実行できず，データの読み出しを待つ間にストールします．