第3部

第3章 普通はプログラマから見えない裏側を Cプログラムを工夫して調査する

キャッシュの細かい挙動を探る

藤井 裕也

キャッシュ・メモリ（以降，キャッシュ）はCPUやGPUの処理性能に大きく影響します．プログラマはキャッシュを有効に使うためにキャッシュ・サイズを考慮しながらデータ構造やループ構造を工夫しますが，その際キャッシュの実装方法まではあまり気にする必要はありません．

しかし，例えばキャッシュが間違った値を返してくるような奇妙な現象に遭遇したとき，キャッシュにどのような実装方法があり，挙動の違いが現れるのは，どういう場合かを知っているとデバッグで有利になるかもしれません．

本稿では筆者が遭遇したトラブルとその調査実験を元に，データ・キャッシュの挙動がキャッシュの実装方法によってどのように変わるかを実験しながら説明します．使用したハードウェアはラズベリー・パイです．原理はどのCPU/GPUでも同じなので，他のハードウェアを使う場合にも参考になるでしょう．

メイン・メモリの遅さを補うキャッシュ（キャッシュ・メモリ）

キャッシュとは，CPU/GPUとメイン・メモリの間に挟まって動作するものであり，高速動作可能なデータの一時置き場です．アクセス頻度の高いデータや，使用するデータ周辺の連続したデータを置いておくことで，毎回メイン・メモリまでデータを取りに行く必要がなくなり，プログラムの実行速度が向上します．

キャッシュには役割ごとに種類があり，プログラム本体を置いておく命令キャッシュと，データを置いておくデータ・キャッシュに分かれている構成が典型的です．

キャッシュは階層構造になっていることが多く，CPU/GPUに近い順にL1，L2，L3と呼ばれます．数字が小さいほど容量が小さく高速，数字が大きいほど容量が大きく低速という設計です．L2以降は命令とデータを区別せず一緒に扱うケースもあります．チップによってはL3がなかったり，L4まであったり，L1とCPUの間にさらに非常に小さいL0[注1]があったりします．

valid	タグ	データ

図1　キャッシュ上のデータとメイン・メモリ上のデータとを結びつけるためタグを使う

キャッシュの基礎知識

● 基本的な仕組み

キャッシュは**図1**のようなデータ構造の集まりです[注2]．

データは，メイン・メモリからコピーした内容そのものです．データはキャッシュ・ラインとも呼ばれ，一度のメモリ・アクセスでこのデータ量が一気にキャッシュに乗ります．

タグはそのキャッシュ・ラインがメイン・メモリのどのアドレスのデータを記録しているのかを表します．タグに必要なビット数はメイン・メモリとキャッシュ・メモリのサイズから決まります．

validはそのキャッシュ・ラインの内容が有効かどうかを示します．CPUがメイン・メモリにアクセスすると，キャッシュの該当ラインのvalidビットが1になり，タグとデータが更新されます．キャッシュのサイズはメイン・メモリに比べると小さいので，全てのデータを乗せることはできません．どのデータをどこに乗せるかはキャッシュの戦略によって変わります．

メイン・メモリ上に**図2**のようなデータがある場合に，キャッシュがどのように動作するのかを説明します．

● 戦略1：ダイレクト・マップ方式

ダイレクト・マップという単純な方式ではアドレスによってどのキャッシュ・ラインが更新されるかが決まっています．キャッシュ・ライン数が4，1つの

注1：単に小さいキャッシュではなく，デコード後の命令を置くためのキャッシュを指すこともあります．

注2：キャッシュからメイン・メモリに書き戻す方法は幾つかあり，方式によってはdirtyビットという情報を持っているものもあります．