計算機結構 期末專題 理學院學士班 張烱郁 105020025

1. 概論

此期末專題的目標在於最佳化取cache index的方式，最直覺的方式是以最低位數(LSB)作為index bit，本專題將以C++語言模擬的方式，比較LSB與另一種取cache index位元的方式，並以miss count作為評斷的標準。

1. 程式結構

本程式以C++撰寫。cache\_set\_LRU為一類別，模擬採用LRU placement policy的cache中的一個set，利用成員函數cache\_set\_LRU :: access來對此cache set進行存取。以vector串聯多個cache\_set\_LRU來模擬一個完整的cache。

1. 演算法及流程

在選取indexing bits時，使用*Zero Cost Indexing for Improved Processor Cache Performance,TONY GIVARGIS*中，3.3 Heuristic Algorithm，流程如下：

Use\_bit陣列紀錄那些bits要做為index

計算第i位元及第j位元在各地址中相同次數和不相同次數的比值

Cij = min(Eij,Dij)/max(Eij,Dij)

存在C二維陣列

計算第i位元在各地址中0、1數量的比值

Qi = min(Zi,Oi)/max(Zi,Oi)

存在Q陣列

initialize

選到所需index bit 的數量

從Q中選出最大者，其index k做為caching index bit

將use\_bit[k]設為1

尚未選到所需index bits的數量

對Q陣列更新

Qi = Qi\*Cki

完成以上流程就可知道此演算法要哪些bits作為index。

不論LSB或Heuristic Algorithm，都使用LRU replacement policy，將index bits 轉換成十進位，即會對應到(二、)中所提之模擬cache的set index，每次存取地址的index被計算出來後，就可以去access其對應到的set。

1. 比較結果

下表是以LSB取index和以Heuristic Algorithm最佳化後取index，跑測試檔案的miss count，紅字表示比起LSB有所提升的部分，其餘則是持平或表現較差。

1. 分析

雖然此演算法造成某些狀況下，miss count反而增加，但我認為並不要緊，因為實際設計硬體前，可以先以軟體模擬幾種不同的演算法，針對不同功能的裝置，採用不同的演算法，因此，只要某演算法可以提升某種狀況下的performance，就有其價值，即便在其他狀況下表現比較差也無妨，改用其他演算法就好。

此結果其實也可以發現，LSB雖然簡單，但也不是省油的燈，表現相當不賴，這是合理的結果，因為採用MSB作為tag，spatial locality的優勢更能夠發揮。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *LSB* | CacheA | CacheB | CacheC | CacheD | CacheE | CacheF |
| DataReference\_real\_up | 24 | 21 | 22 | 21 | 21 | 21 |
| DataReference\_n\_comp | 96 | 27 | 30 | 27 | 28 | 27 |
| DataReference\_n\_real | 89 | 23 | 26 | 23 | 24 | 23 |
| InstReference\_iir\_one | 118 | 89 | 105 | 89 | 90 | 89 |
| InstReference\_lms | 113 | 86 | 99 | 86 | 92 | 86 |
| InstReference\_matrix | 123 | 94 | 110 | 94 | 96 | 94 |
| InstReference\_n\_comp | 119 | 92 | 107 | 92 | 103 | 92 |
| InstReference\_n\_comp | 3000 | 3000 | 3000 | 3000 | 3000 | 3000 |
| *Optmized* | CacheA | CacheB | CacheC | CacheD | CacheE | CacheF |
| DataReference\_real\_up | 22 | 21 | 21 | 21 | 21 | 21 |
| DataReference\_n\_comp | 249 | 27 | 79 | 27 | 31 | 27 |
| DataReference\_n\_real | 26 | 23 | 26 | 23 | 27 | 23 |
| InstReference\_iir\_one | 131 | 89 | 109 | 89 | 91 | 89 |
| InstReference\_lms | 112 | 86 | 107 | 86 | 97 | 86 |
| InstReference\_matrix | 122 | 94 | 101 | 94 | 103 | 94 |
| InstReference\_n\_comp | 187 | 92 | 108 | 92 | 104 | 92 |
| InstReference\_n\_comp | 3000 | 3000 | 3000 | 3000 | 3000 | 3000 |

1. 參考資料

Zero Cost Indexing for Improved Processor Cache Performance

TONY GIVARGIS

University of California, Irvine