COP 5405: Advanced Algorithms

Fall 2006

Lecture 15

Derivation of Cache Complexity for Cache Oblivious Matrix Multiplication

When using a divide and conquer approach, we can write *C* as

$$C = \begin{matrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{matrix}$$

Thus, we recursively solve eight sub-problems of half the size in order to compute *C*.

Note that if each of the three submatrices is of size $n' \times n'$, where $n' \leq (⅓\, Z)^{0.5}$, then they will all fit in cache, and in the recursive calls on their submatrices, each data item will be brought into cache only once. We can, therefore, write the following expression for the cache complexity.

$$Q(n) \leq \begin{matrix} \theta(3n^2/L), & n \leq (⅓\, Z)^{0.5} \\ 8Q(n/2), & otherwise \end{matrix}$$

For large n, we get $Q(n) = 2^3Q(n/2) = 2^6Q(n/4) = \ldots = 2^{3i}Q(n/2^i)$. If $n/2^i \leq (⅓\, Z)^{0.5}$, then we know the solution explicitly. For this value of $n/2^i$, $2^i = n\ (⅓\, Z)^{-0.5}$. Therefore, $Q(n) = n^3\ (⅓\, Z)^{-1.5}Q((⅓\, Z)^{0.5}) = n^3\ (⅓\, Z)^{-1.5}\theta\,(3(⅓\, Z)/L) = \theta(n^3Z^{-0.5}/L)$.