Tuning Libraries to Effectively Exploit Memory

Prof. Misha Kilmer
Emily Reid
Stacey Ecott
A Project in Numerical Linear Algebra

- An understanding of mathematics (linear algebra)

- An understanding the movement of data in the computer memory to construct efficient algorithms for solving large-scale linear systems of equations where the matrices are sparse (have lots of zero entries)
Storage of Arrays

- **Row-wise**
  - A(1,1) → a(1)
  - A(2,1) → a(2)
  - A(3,1) → a(3)
  - A(1,2) → a(4)
  - A(2,2) → a(5)
  - A(3,2) → a(6)
  - A(1,3) → a(7)
  - A(2,3) → a(8)
  - A(3,3) → a(9)

- **Column-wise**
  - A(1,1) → a(1)
  - A(2,1) → a(2)
  - A(3,1) → a(3)
  - A(1,2) → a(4)
  - A(2,2) → a(5)
  - A(3,2) → a(6)
  - A(1,3) → a(7)
  - A(2,3) → a(8)
  - A(3,3) → a(9)
The idea is to isolate frequently occurring code into subprograms where it can be optimized…

BLAS: basic linear algebra subprograms

Original dot code:
- \( i = 1 \)
- for \( k = 1 \) to \( n \)
  - \( x(k) = b(k) \)
  - for \( j = 1 \) to \( k-1 \)
    - \( x(k) = x(k) - l(i) \times x(k) \)
    - \( i = i + 1 \)
  - end for \( j \)
- \( x(k) = x(k)/l(i) \)
- \( i = i + p - k \)
- end for \( k \)
Dot code and new version of program:

- `dot(n,x,y)`
- `s = 0`
- `for k = 1 to n`
  - `s = s + x(k)*y(k)`
- `end for k`
- `return s`
- `end dot`

- `for k=1 to n`
  - `x(k) = (b(k) – dot(k-1, L(k-1), x(1)))/A(k,k)`
- `end for k`
Algorithm

Assuming \( n, m, \) and \( k \) are divisible by 2, matrices can be partitioned into blocks such that a matrix consists of blocks \( A_{11}, A_{12}, A_{21}, A_{22} \). Then, when multiplying matrices \( A \) and \( B \) to get matrix \( C \), the upper-left hand block of \( C \) is \( A_{11}B_{11} + A_{12}B_{21} \).
Localities

- **Locality in Time** - The concept that a resource that is referenced at one point in time will be referenced again sometime in the near future.

- **Locality in Space** - The concept that likelihood of referencing a resource is higher if a resource near it was just referenced.

- **Cache Coherency** - The concept that memory is accessed sequentially from the cache.
Method 3

\[ y = 0.0015x^{3.4114} \]

\[ R^2 = 0.9994 \]

Method 3-2

\[ y = 0.002x^{3.3946} \]

\[ R^2 = 0.9989 \]
Problems

- Memory access faults with sufficiently large matrices, potentially due to algorithm.
- Relatively small variance in timing, presumably due to other processes on server
Future Work

- Test algorithm on specific matrix sizes, specifically “skinny” matrices
- Apply current algorithm to sparse matrices
- Test with blocking
- Determine better algorithm for sparse matrices
Thank You!!!