Modeling loop performance
Loop transformations

- Many kinds of loop transformations
  - Loop permutation/interchange
  - Loop blocking/tiling
  - Loop reversal
  - Loop fusion
- Want to understand the effects of these transformations
  - How does a transformation impact performance?
  - Can we predict this impact?
- Focus on a case study: matrix-matrix multiply and loop interchange
Matrix-matrix multiply

- Key kernel in linear algebra
- How much data? How much computation?
- Significant data reuse
- Important factor in performance: miss ratio
- Does miss ratio depend on problem size?
- Interesting fact: can execute loops in any order
  - Does miss ratio depend on loop order?
- Can we predict miss ratio?

\[
\begin{align*}
  &\text{for } i \in [0 : 1 : N - 1] \\
  &\quad \text{for } j \in [0 : 1 : M - 1] \\
  &\quad \quad \text{for } k \in [0 : 1 : K - 1] \\
  &\quad \quad \quad C_{ij} = C_{ij} + A_{ik} \times B_{kj}
\end{align*}
\]
Miss ratios

Tuesday, February 2, 2010
Explaining miss ratios

• When matrices are small, everything fits in cache
  • Only get cold misses, no capacity misses
  • Misses: $3N^2/b$, accesses: $4N^3$ (why?)
  • Miss rate: $3/(4bN)$
• Miss rate goes down as problem size goes up!

• How long does this happen?
  • Naive guess: happens as long as all three matrices fit in cache ($N \leq \sqrt{C/3}$)
  • On Itanium: should happen when $N \leq 104$
Miss-rate regimes

- When a matrix fits entirely in cache, it experiences temporal and spatial locality
  - Misses: $N^2/b$
- If a matrix is being walked in row-major order, it may experience spatial locality, but not temporal locality
  - Only get a miss 1 out of b accesses
  - Misses: $N^3/b$
- Other times, a matrix experiences no locality
  - Every access misses
  - Misses: $N^3$
- (What about a matrix that only experiences temporal locality?)
Predicting miss rates

- To predict a miss rate, we need to determine, for each matrix:
  - Whether it experiences no locality, spatial locality, or both spatial and temporal locality
  - At which point the matrix transitions between the various regimes
Stack distance

- Introduced by Mattson et al. in 1970
- The stack distance of a memory location is the number of distinct cache lines touched between successive accesses to that location
  - Called stack distance because it can be calculated with a reuse stack
  - Also called reuse distance
Stack distance

- We introduce two types of stack distance:
  - $d_t(M)$: the stack distance between successive accesses to a given element of $M$
  - $d_s(M)$: the minimum stack distance between successive accesses to distinct elements of $M$ that lay on the same cache line
- If $C < b^*d_t(M)$, matrix does not have temporal locality
  - By the time we touch the same element again, we’ve brought in too many other elements into cache
- If $C < b^*d_s(M)$, matrix does not have spatial locality
Computing $d_t$ and $d_s$

- A is walked in row major order in inner loop
- $d_s(A) = 3$ (why?)
- Note: $d_s$ is not dependent on $N \to A$ always has spatial locality
- How many iterations does it take to return to the same element of $A$?
- What is $d_t(A)$?

\[
\begin{align*}
&\text{for } i \in [0 : 1 : N - 1] \\
&\text{for } j \in [0 : 1 : M - 1] \\
&\text{for } k \in [0 : 1 : K - 1] \\
&C_{ij} = C_{ij} + A_{ik} \ast B_{kj}
\end{align*}
\]
Miss rates for B and C

- What is $d_t(B)$?

- What is $d_s(B)$?

- What about for C?

\[
\begin{align*}
\text{for } i & \in [0:1:N-1] \\
\text{for } j & \in [0:1:M-1] \\
\text{for } k & \in [0:1:K-1]
\end{align*}
\]

\[
C_{ij} = C_{ij} + A_{ik} \times B_{kj}
\]
Putting it all together

\[ miss_{ijk,A}(N, b, C) = \begin{cases} 
  \frac{N^2}{b} & \text{if } bN + N \leq C \\
  \frac{N^3}{b} & \text{otherwise}
\end{cases} \]

\[ miss_{ijk,B}(N, b, C) = \begin{cases} 
  \frac{N^2}{b} & \text{if } (N^2 + 2N) \leq C \\
  \frac{N^3}{b} & \text{if } bN + N \leq C \\
  N^3 & \text{otherwise}
\end{cases} \]

\[ miss_{ijk,C}(N, b, C) = \frac{N^2}{b} \]
Putting it all together

\[
\text{miss}_{ijk}(N, b, C) = \begin{cases} 
3N^2/b & (N^2 + 2N) \leq C \\
N^3/b + 2N^2/b & bN + N \leq C \\
(b + 1)N^3/b & \text{otherwise}
\end{cases}
\]

\[
\text{ratio}_{ijk}(N, b, C) = \begin{cases} 
3/(4bN) & N \leq \sqrt{(C')} \\
1/(4b) & N \leq C/(b + 1) \\
(b + 1)/(4b) & \text{otherwise}
\end{cases}
\]
Miss ratios for other orders

\[
\text{ratio}_{ijk}(N, b, C) = \begin{cases} 
3/(4bN) & N \leq \sqrt{C} \\
1/(4b) & N \leq C/(b + 1) \\
(b + 1)/(4b) & \text{otherwise}
\end{cases}
\]

\[
\text{ratio}_{jki}(N, b, C) = \begin{cases} 
3/(4bN) & N \leq \sqrt{C} \\
1/(4b) & N \leq C/(2b) \\
1/2 & \text{otherwise}
\end{cases}
\]

\[
\text{ratio}_{kij}(N, b, C) = \begin{cases} 
3/(4bN) & N \leq \sqrt{C} \\
1/(4b) & N \leq C/2 \\
1/(2b) & \text{otherwise}
\end{cases}
\]
Performance regimes

- **Large-cache regime**
  - All matrices exhibit temporal and spatial locality
  - Miss rate decreases as matrix size gets larger
  - For all orders, occurs until $N \geq \sqrt{C}$

- **Medium-cache regime**
  - One matrix starts incurring capacity misses
  - Others still enjoy locality

- **Small-cache regime**
  - Two matrices start suffering capacity misses
Predicting performance

- Itanium 2 architecture:
  - Line size: 16 doubles
  - Cache size (L2): 32K doubles

\[ \text{ratio}_{ijk}(N, b, C) = \begin{cases} 
  3/(4bN) & N \leq \sqrt{(C')} \\
  1/(4b) & N \leq C/(b + 1) \\
  (b + 1)/(4b) & \text{otherwise} 
\end{cases} \]

- Predicted miss rates:
  - decrease while \( N \leq 181 \)
  - 1.5625% while \( N \leq 1927 \)
  - 26.5625% afterwards
Miss ratios

L2 Miss Rate (%) vs. N (elements)

- k inner
- i inner
- j inner

Tuesday, February 2, 2010
Loop permutation

- Loop permutation clearly an important transformation
  - Can lead to massive performance improvements
- How do we determine when loop permutation is legal?
- How do we automatically generate permuted code?
  - Straightforward for some loops (like MMM)
  - Much harder for other loops
- How do we know if loop permutation will be useful?
  - Don’t want to change $ijk$ loop into $jki$ loop!
- Are there other transformations we might want to perform?
- The next set of lectures will answer these questions