

CS267: Matrix Multiplication

Marghoob Mohiyuddin
marghoob@eecs.berkeley.edu

Introduction

- Any questions?
 - Topics you want to see?
- Opteron memory system
- Compiler optimizations
- Hardware counters
- Automatic tuning

Opteron Memory System

- Two level cache memory hierarchy
- Two level Translation Lookaside Buffer

What are Caches?

- Predict what data the processor will need and have it close to the processor so that memory accesses are quick
- Two ways of predicting
 - Temporal locality
 - You are likely to reuse data
 - Spatial locality
 - You are likely to use data adjacent to recently used data

Virtual Memory System

- Operating systems partition memory space into fixed size units called pages and deal with them
- To protect applications from each other, use a level of indirection
 - Securely translates addresses so that mapping prevents any possible interference
 - Sandbox various applications from each other

Virtual Memory System (contd.)

- Translation is potentially very expensive
 - Can be in milliseconds if your machine is overstressed
 - 1 ms on a 1 GHz machine is 1 million instructions!
- Use a translation cache (TLB) to speed this up
 - Typically you don't see the effects
 - Unless you step right over it

When Caches do Well

- If you increase your reuse
- If you make sure to access adjacent elements
- Everything Patterson and Hennessey tell you about the greatness of caches

When Caches and TLBs do Badly

- Accessing more data elements than there is storage for
- If you only use part of a cache line
 - Arise from small strides in the accesses
 - $1 < \text{stride} < 16$
- If you only use a fraction of the cache lines
 - Arises from large strides in the accesses
 - $16 < \text{stride} < 256$
- Overstress the TLBs
 - Arises from huge strides in the accesses
 - $\text{strides} > 256$

Ways Around it

- Make sure not to overstep the amount of data
 - Use blocking
 - Maybe consider multiple levels of blocking
 - Realize that you need storage for three matrices
 - Operands and result
 - Mind the TLB

Ways Around it (contd.)

- Cache padding
 - Power of 2 strides can potentially cause more collisions and artificially decrease the size of a cache
- Example
 - Lets say cache line size is 4 ints
 - Lets say we only have 32 lines

```
for(i=0; i<N; i+=16) {  
    x[i][0] += 4;  
}
```

- Separate instruction and data L1 caches
 - Size: 64 KB each
 - Line size: 64 bytes
 - Number of lines: 1024
 - Associativity: 2-way
- Exclusive L2 unified cache
 - Size: 1 MB
 - Line size: 64 bytes?
 - Number of lines: 16384
 - Associativity: 16-way

Virtual Memory Info

- Separate instruction and data level 1 TLBs
 - 40 entry, fully associative
- Unified Level 2 TLB
 - 512 entry, 4-way set associative

- Loop unrolling
 - 2 floating point units which can run in parallel
 - Do multiple iterations of a loop simultaneously

Example: Dot Product

- Do $result = aX \cdot Y$

```
for( i=0; i<N; i++ ) {  
    result += a*X[i]*Y[i];  
}
```

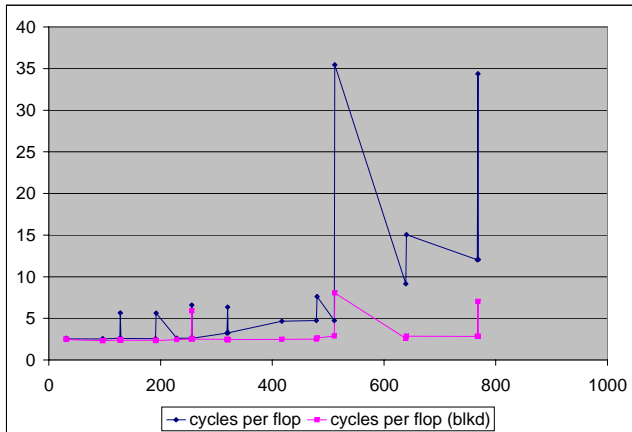
- Slightly better

```
for( i=0; i<N; i+=4 ) {  
    result_a += a*X[i]*Y[i];  
    result_b += a*X[i+1]*Y[i+1];  
    result_c += a*X[i+2]*Y[i+2];  
    result_d += a*X[i+3]*Y[i+3];  
    result += result_a+result_b+result_c+result_d;  
}  
/*any cleanup code because N is not multiple of 4*/
```

Hardware Counters

- Switch over to code to show example
- They are useful
- Allow you to make analytical decisions about whether an optimization is working

Performance: Naive Code vs. Blocked Code



A Better Way?

- Automatic tuning
 - Make your code generic
 - Automatically generate variants
 - Generate loop unroll of 1 to 12 for dot product case
 - Iterate over your implementation space
 - Watch a movie
 - Come back and just pick the implementation that did the best and say its the winner

Pros and Cons

- Pros
 - A script does most of your work while you can spend your time doing other things
 - Writing a PhD thesis or watching a movie or just getting a life
- Cons
 - Often hard to generalize (not a problem with DGEMM)
 - Dont get as good performance as hand tuned versions (but you can get very close)
 - Can take a long time if your search space is huge

Automatic Tuning Hints

- Most of the optimizations are independent of each other
 - Fix all but one parameter and then find the best value
 - Prune your search space to be sane
 - Leverage PAPI to see if you are pruning the search space correctly

Questions?

- Fire them!