Why Not Just Use Python?

Q: Couldn’t I just use Python?
A: Yes, but … in this class, you’ll learn how to do better.

Q: How much better?
A: Let’s look at some data from Saman Amarasinghe (and Martin Rinard and Charles Leiserson) from MIT, c. 2009.

The Problem: Dense Matrix Multiply

One you know and love: dense matrix multiply!

1024×1024 matrices
2³⁰ multiply-adds
dual quad-core Intel machines

if you … you lose
ignore processor parallelism 3.5×
don’t use Intel hand-optimized assembly library 2.7×
don’t bother to vectorize (MMX/SSE) 2.8×
ignore cache size 1.7×
ignore data org. in memory (don’t transpose matrix) 3.4×
use Java! 2.1×
use objects 2.2×
allow double & integer matrices 2.4×
use immutable objects! 220×

 languages … sort of 3000000 1

° note that 2.1× is the kind of number that managed language proponents claim
° the real cost is having no way to get at the remaining 100×, even if you’re
° [MMX = multimedia extensions, SSE = streaming SIMD extensions]
Objective

- To understand the organization of memory based on dynamic RAM (DRAM).
- To understand the use of burst mode and multiple banks (both sources of parallelism) to increase DRAM performance (data rate).
- To understand memory access coalescing, which connects GPU kernel performance to DRAM organization.

Most Large Memories Use DRAM

- **Random Access Memory (RAM):** same time needed to read/write any address
- **Dynamic RAM (DRAM):**
  - bit stored on a capacitor
  - connected via transistor to bit line for read/write
  - bits disappear after a while (around 50 msec, due to tiny leakage currents through transistor), and must be rewritten (hence dynamic)

Global Memory (DRAM) Bandwidth

Ideal

Reality

Many Cells (Bits) per Bit Line

- About 1,000 cells connect to each BIT LINE.
- Connection/disconnection depends on SELECT line.
- Some address bits decoded to connect exactly one cell to the BIT LINE.
DRAM is Slow But Dense

- Capacitance... 
  - tiny for the **BIT**, but 
  - huge for the **BIT LINE**
- Use an amplifier for higher speed!
- Still **slow**...
- But only need 1 transistor per bit.

**Sense Amps**

---

DRAM Bank Organization

- SELECT lines connect to about 1,000 bit lines.
- Core array has about O(1M) bits
- Use more address bits to choose bit line(s).

---

DRAM Interfaces are Clocked

- DRAM **cells** are not clocked (clocking requires transistors).
- DRAM **interfaces** are clocked.
  - DDR: Core speed = ½ interface speed
  - DDR2/GDDR3: Core speed = ¼ interface speed
  - DDR3/GDDR4: Core speed = ⅛ interface speed
  - ... likely to be worse in the future
Modern DRAM systems are designed to be always accessed in burst mode. Burst bytes are transferred but discarded when accesses are not to sequential locations.
A Simple Matrix Multiplication Kernel (review)

```c
__global__ void MatrixMulKernel(float* M, float* N, float* P, int Width)
{
    // Calculate the row index of the P element and M
    int Row = blockIdx.y * blockDim.y + threadIdx.y;
    // Calculate the column index of P and N
    int Col = blockIdx.x * blockDim.x + threadIdx.x;
    if ((Row < Width) && (Col < Width)) {
        float Pvalue = 0;
        // each thread computes one element of the block sub-matrix
        for (int k = 0; k < Width; ++k)
            Pvalue += M[Row*Width+k] * N[k*Width+Col];
        P[Row*Width+Col] = Pvalue;
    }
}
```

Two Access Patterns

(a) M[Row*Width+k]  N[k*Width+Col]  k is loop counter in the inner product loop of the kernel code,
N accesses are coalesced.

M accesses are not coalesced.

Use shared memory to enable coalescing in tiled matrix multiplication

ANY MORE QUESTIONS?
READ CHAPTER 5