Parallel Architecture

Announcements

- The RamCT merge is done! Please repost introductions.
- Manaf's office hours
- HW0 is due tomorrow night, please try RamCT submission
- HW1 has been posted

Today

- Isoefficiency
- Levels of parallelism in architecture and programming models
- Leveraging the memory hierarchy
- Understanding performance limits
  - Machine balance
  - Roofline model
- Performance analysis logistics for this semester

Recall Parallel Performance Metrics

Speedup

$T_s(N)$ exec time for efficient serial computation on problem size N

$T(N,P)$ exectime for parallel version of computation on problem size N with P processors

speedup is the serial exec time divided by the parallel exec time

$$S = T_s(N)/T(N,P)$$

Efficiency

efficiency is the percentage of all the processing power that is being used

$$E = T_s(N)/(PT(N,P)) = S/P$$

Isoefficiency

Main idea

–Keep the parallel efficiency the same and increase the problem size.

–The isoefficiency function indicates how much the problem size needs to increase as processors are added.

Example $\sum_{i=0}^{N} A[i]$

–Summation is a reduction

–Serial execution time $T_S(N) = O(N)$

–Parallel execution time needs to include communication time and idle time, or parallel overhead $T(N,P) = O(N/P + log_2(P))$

–Isoefficiency function indicates how the problem size must increase as a function of the number of processors to maintain the same efficiency.

$$N = P \lg P$$

Parallel Architectures

Parallelism

- Many processing units (floating point and integer units, processors)
- Places to store data (memory, cache)
- Various ways the processors are connected to the memory.
- No real "winner" parallel architecture, so variety in programming models.
- Tradeoffs between portable code and efficient code.
- Goal of automation tools (e.g., compilers) is to find the sweet spots

Levels of Parallelism in Architectures

Instruction Level Parallelism

- Pipeline parallelism
- superscalar
- VLIW (very long instruction word)
- Vector processing units

Shared Memory Parallelism: multiple processors connected to same memory usually with cache coherency

- Multicore machines like veges
- Node of the cray
- Shared memory for a thread block in a GPU and global memory in GPU

Distributed Memory Parallelism: multiple processors each with own memory connected with an interconnect

- Between nodes of the cray
- Clusters

Levels of Parallelism in Programming Models

Instruction Level Parallelism

- Mostly handled by the compiler
- Loop unrolling
- MMX, SSE, and AVX

Shared Memory Parallelism

- OpenMP
- Threads: pthreads, Java threads, Cilk, TBB

Distributed Memory Parallelism

- MPI
- PGAS languages
  - Co-array Fortran, Unified Parallel C, Titanium, …
- New languages with concept of places/locales
  - X10
  - Chapel

Memory Hierarchy

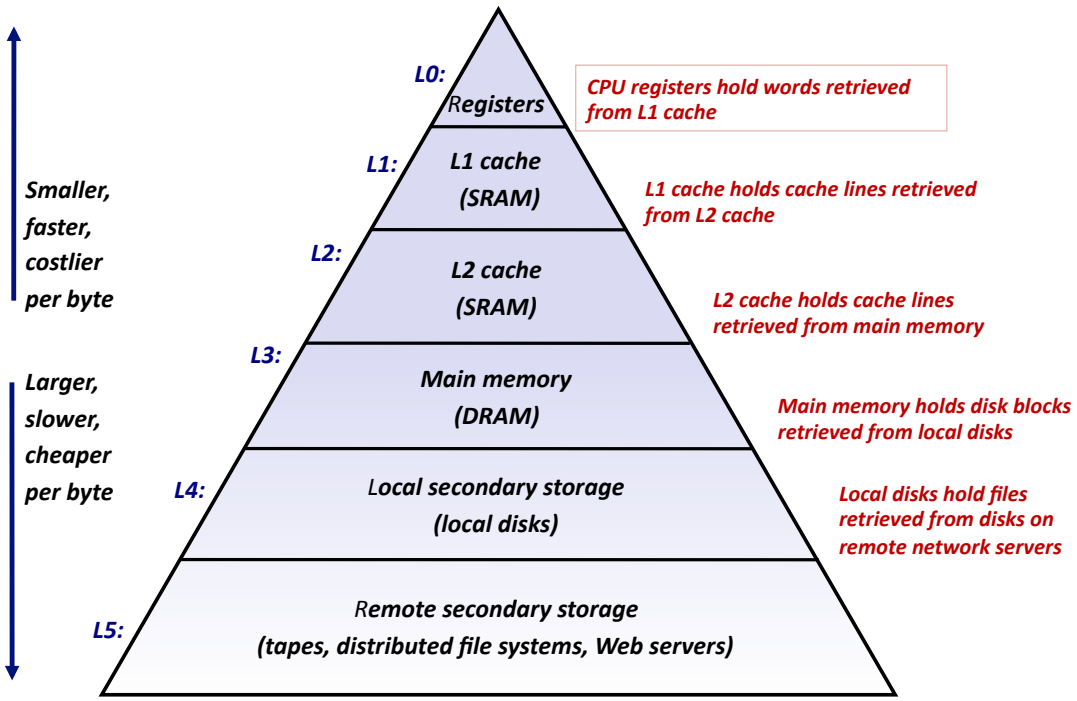
What?

- There is an ever growing hierarchy of caches that lie between main memory and the processing unit. Eg, L1 cache, L2 cache, L3 cache
- In parallel machines the hierarchy causes non-uniform memory access (NUMA) due to subsets of cores sharing caches.

Why?

- It takes 100-1000 cycles to access main memory directly.
- Caches (SRAM-based memory) are fast but expensive and therefore not that large.

An Example Memory Hierarchy



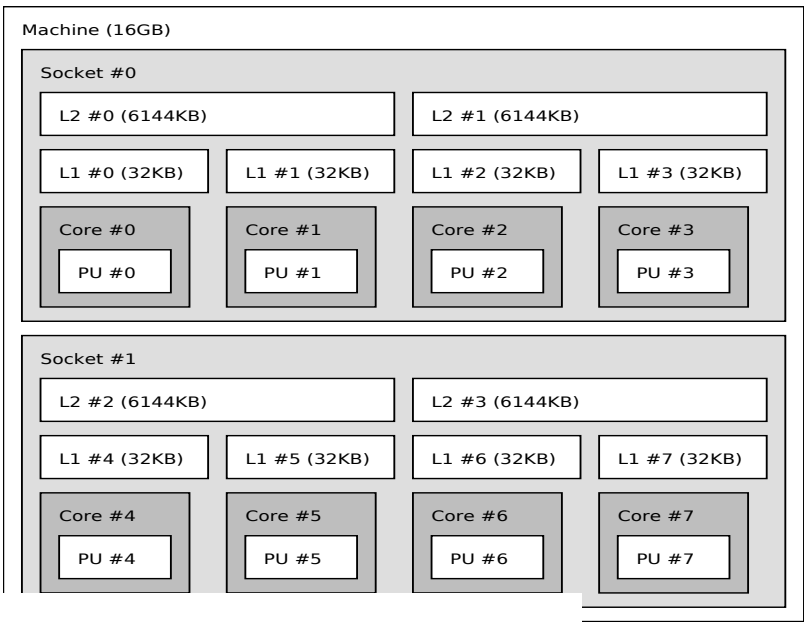
Source: http://www.cs.cmu.edu/afs/cs/academic/class/15213-f10/www/lectures/09-memory-hierarchy.pptx

Examples of Caching in the Hierarchy

Cache Type	What is Cached?	Where is it Cached?	Latency (cycles)	Managed By
Registers	4-8 bytes words	CPU core	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware
L1 cache	64-bytes block	On-Chip L1	1	Hardware
L2 cache	64-bytes block	On/Off-Chip L2	10	Hardware
Virtual Memory	4-KB page	Main memory	100	Hardware + OS
Buffer cache	Parts of files	Main memory	100	OS
Disk cache	Disk sectors	Disk controller	100,000	Disk firmware
Network buffer cache	Parts of files	Local disk	10,000,000	AFS/NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

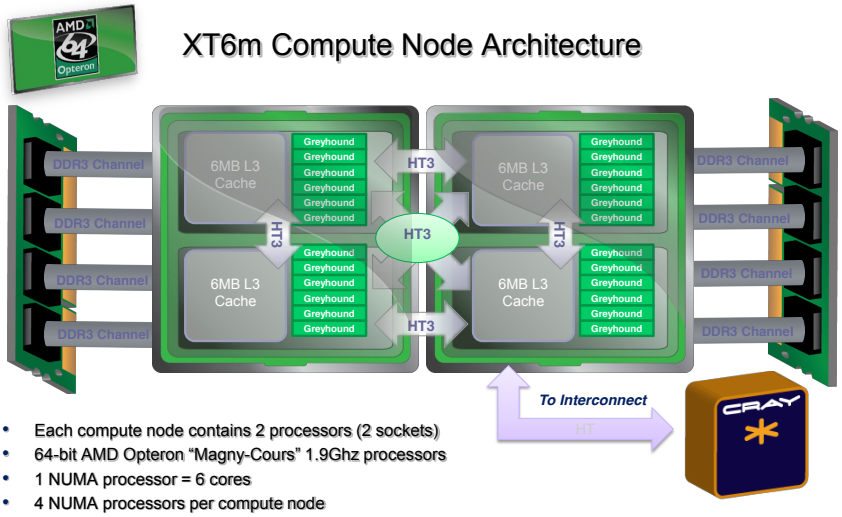
Source: http://www.cs.cmu.edu/afs/cs/academic/class/15213-f10/www/lectures/09-memory-hierarchy.pptx

Harpertown Architecture (veges in department)




```
//commands
setenv PATH /s/bach/e/proj/rtrt/software/bin:$PATH
lstopo --output-format pdf > lstopo-out.pdf
```

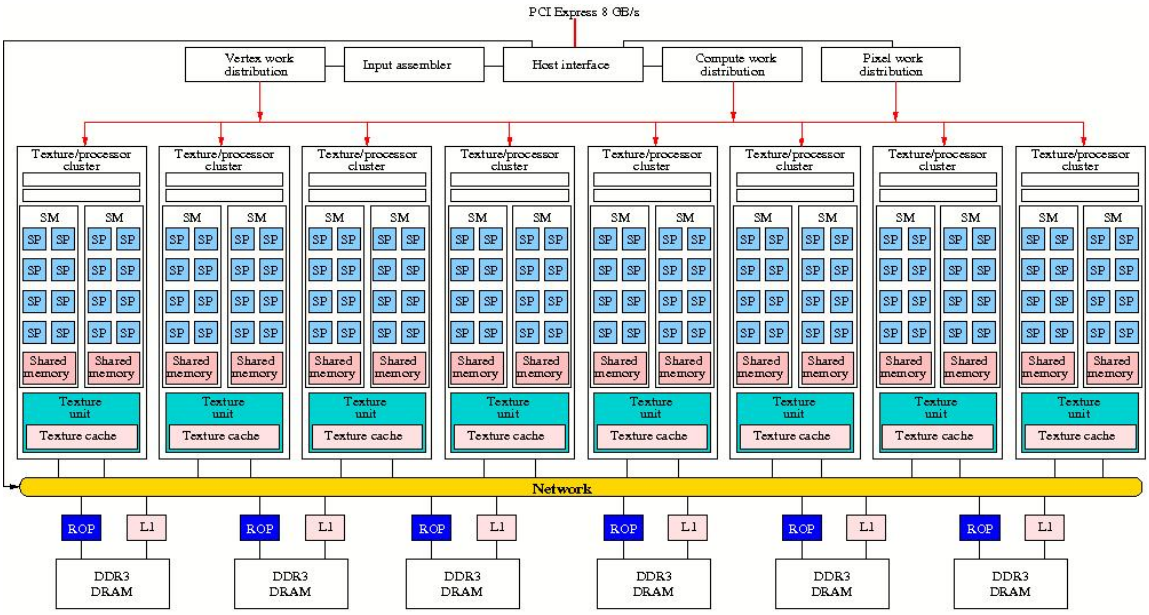
Cray Architecture



- Each compute node contains 2 processors (2 sockets)
- 64-bit AMD Opteron "Magny-Cours" 1.9Ghz processors
- 1 NUMA processor = 6 cores
- 4 NUMA processors per compute node
- 24 cores per compute node
- 4 NUMA processors per compute blade
- 32 GB RAM (shared) / compute node = 1.664 TB total RAM (ECC DDR3 SDRAM)
- 1.33 GB RAM / core

Source: http://istec.colostate.edu/istec_cray/tutorial_3_30_11.pdf

NVIDIA Tesla (bananas, coconuts, apples, and oranges)



SM: Streaming Multiprocessor; SP: Streaming Processor; ROP: Raster Operation Processor

Source: http://www.euroben.nl/reports/web09/tesla.jpg

Data Reuse and Data Locality

Definitions

- *Temporal reuse* is when the same memory location is read more than once.
- *Spatial reuse is* when more than one memory location mapped to the same cache line are used.
- *Temporal and spatial data locality* occurs when those reuses occur before the cache line is kicked out of cache.

Matrix Multiply Example <Draw pictures in class>

```
for (i=0; i<N; i++) {
  for (j=0; j<N; j++) {
    C[i][j] = 0;
    for (k=0; k<N; k++) {
      C[i][j] += A[j][k] * B[k][j]; } } }
```

Sparse Matrix Vector Example <Draw pictures in class>

```
for (j=0; j<N; j++) { Y[i] = 0; }
for (i=0; i<NNZ; i++) {
  Y[row[i]] += val[i]*X[col[i]];
}
```


Arithmetic Intensity and Machine Balance

Arithmetic Intensity

- Ratio of arithmetic operations to memory operations within a computation/loop.

Machine Balance

- Ratio of peak floating point ops per cycle to sustained memory ops per cycle.
- Number of floating point operations that can be performed during the time for an average memory access.

Why?

- If the arithmetic intensity doesn't match the machine balance then the computation will be memory bound.
- If there is data reuse then it might be possible to store data in scalars (i.e. registers) and raise the arithmetic intensity.

Roofline Model (Will be using this in HW2)

Roofline: An Insightful Visual Performance Model for Multicore Architecture

- By Sam Williams, Andrew Waterman, and David Patterson
- ACM Communications, April 2009, Vol 52, No 4.

Operational Intensity: Operations per byte of DRAM traffic.

Roofline graph per machine

- FLOPS/sec versus operational intensity
- Horizontal line for the peak floating point performance (compute bound)
- Diagonal line for the measured peak memory performance (memory bound)

Placing ceilings to represent how performance optimizations can help

- Improve ILP and apply SIMD (computation bound)
- Balance floating point operation mix (computation bound)
- Unit stride accesses (memory bound)
- Memory affinity (memory bound)

Logistics of Performance Analysis

Multiple observations are necessary

- Execution time will not be the same for every run: other users, slightly different cache alignments, etc.
- Need to plot the execution time average of 5-10 observations with bars for standard deviation. Is 5-10 enough?

Performance Issues for specific architectures

- Throughout the semester post programming techniques that improve performance on the veges, cray, and tesla machines.

Concepts

Isoefficiency

Levels of Parallelism in Arch and Programming Languages

- ILP, shared memory, distributed memory
- Loop unrolling, SSE and AVX instructions, do all loops, SPMD, message passing

Memory Hierarchy

- NUMA
- Data reuse and data locality
- Programming constructs to manage data locality?

Performance limits

- Machine balance
- Roofline model (how to draw the graph)
  - operational intensity
  - Programming techniques to break through ceilings

Next Time

Reading

- Berkeley View

Homework

- HW0 is due Wednesday 1/25/12
- HW1 is due Wednesday 2/1/12

Lecture

- Scientific Applications of Interest

Isoefficiency

→ typo on slide
→ ILP on cray?

$P = 16$

$N = 16$

$T_s(N) = 16$

$T_p(N,P) = N/P + \lg P$

$= 1 + 4 = 5$

$$E = \frac{T_s(N)}{P\,T(N,P)} = \frac{16}{16*5} = 20\%$$

Let $N > P$

$N = 16 \quad P = 4$

$T(N,P) = N/P + \lg P$

$= 4 + 2 = 6$

$$E = \frac{16}{4*6} = \frac{16}{24} = 67\%$$

General Case

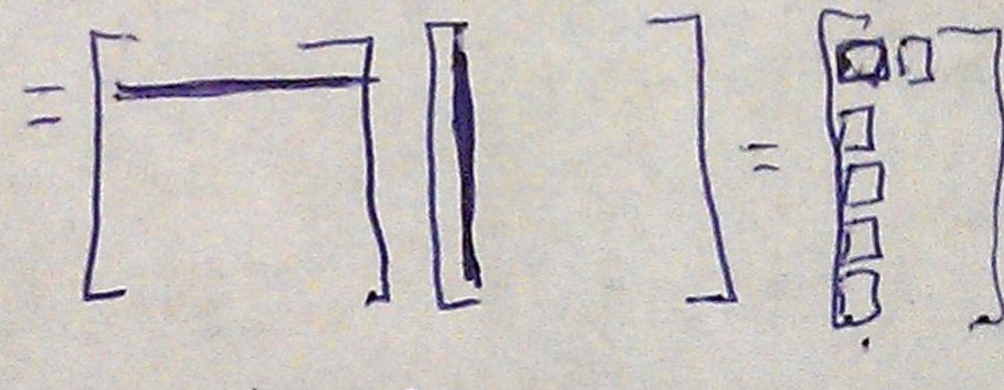
$$E = \frac{}{P\frac{N}{P} + P\lg P}$$

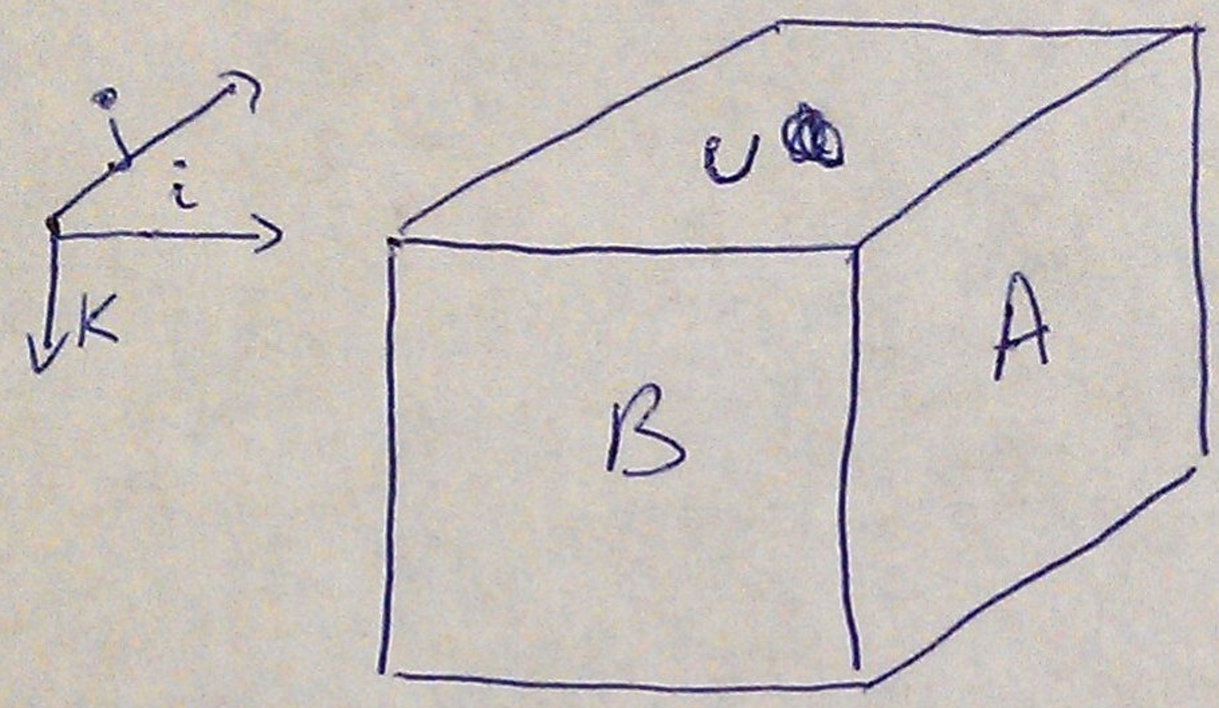
$$= \frac{1}{1 + (P\lg P/N)}$$

$N \geq P \lg P$

Matrix Multiply

$C = AB$



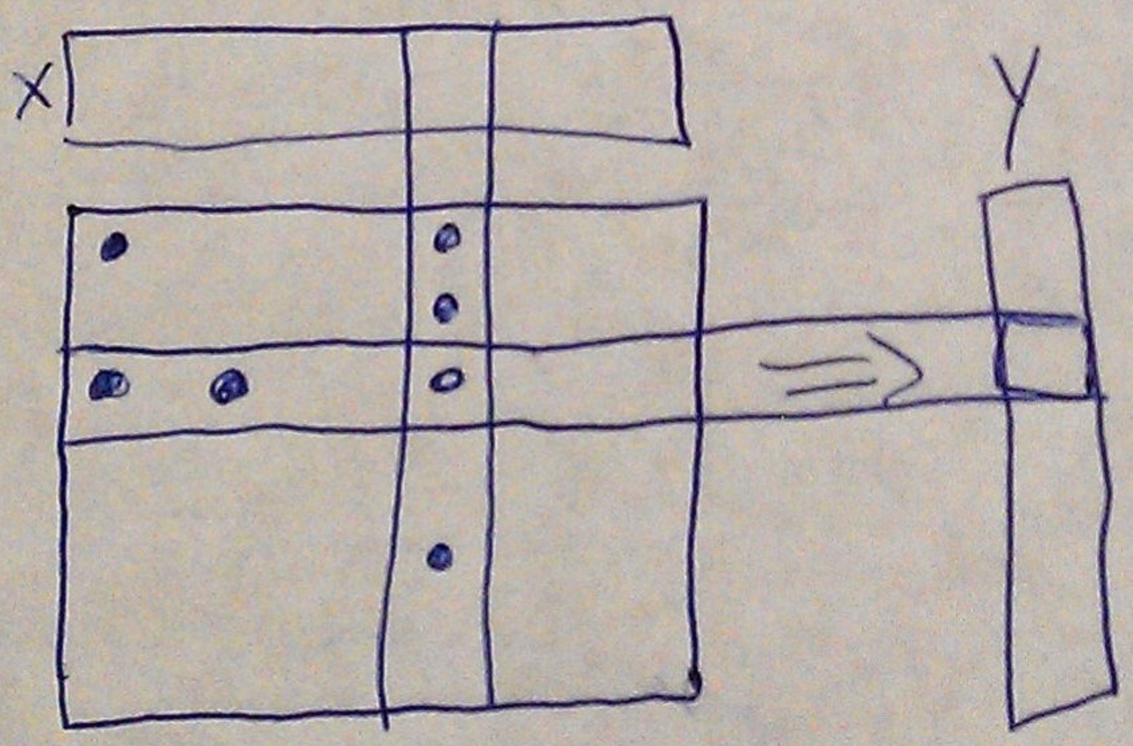


$\frac{O(N^3) \text{ computation}}{O(N^2) \text{ data}}$

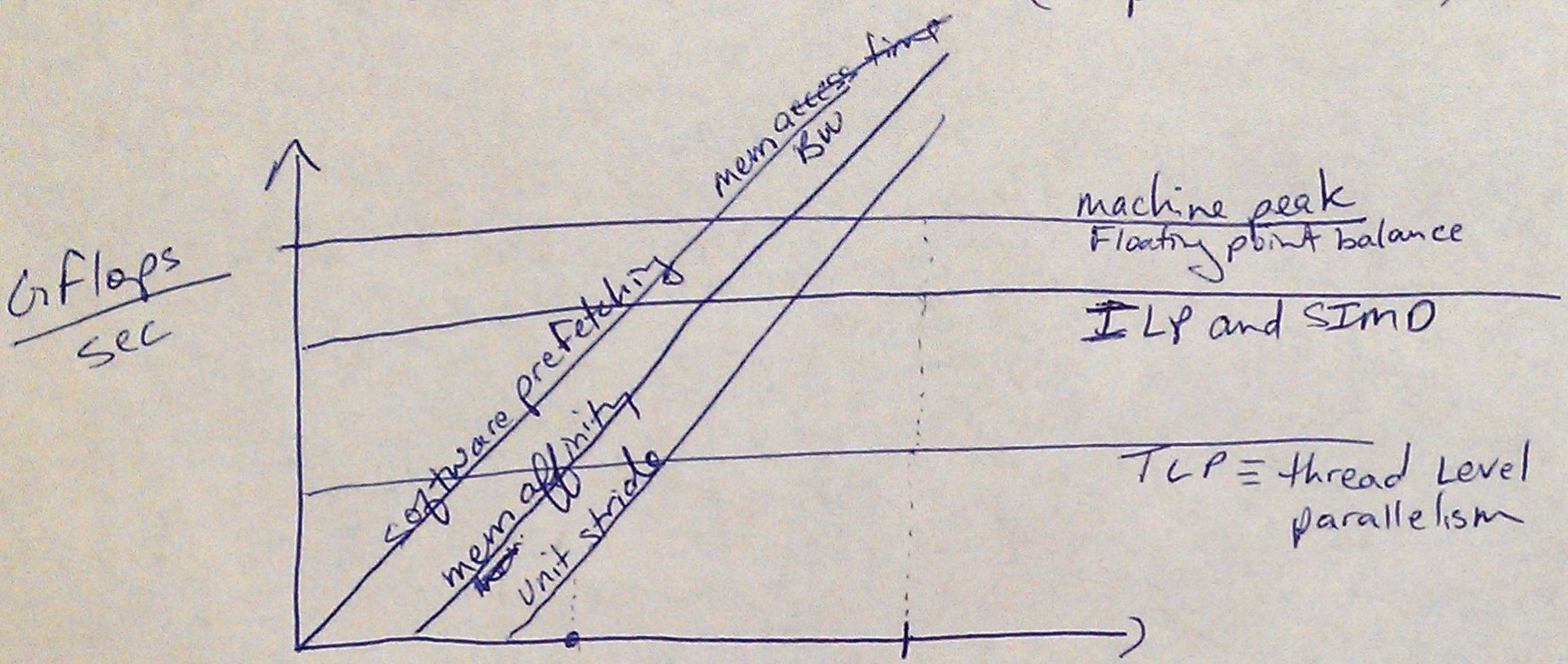
reuse = $O(N)$

SpMV

$y = Ax$



Roofline Model (Opteron X2)



Operational Intensity

$$\frac{\text{GFlops}}{\text{Byte}}$$

$$\frac{\text{GFlops}}{\text{sec}} = \text{Peak Mem BW} * \text{Op Intensity}$$

$$= \frac{\text{Bytes}}{\text{sec}} * \frac{\text{GFlops}}{(\text{Bytes})}$$