CA 714CA Midterm Review
C5 Cache Optimization

• Reduce miss penalty
  – Hardware and software

• Reduce miss rate
  – Hardware and software

• Reduce hit time
  – Hardware

• Complete list of techniques in figure 5.26 on page 499
C5 AMAT

• Average memory access time
  = Hit time + miss rate * miss penalty
  + Useful in focusing on the memory performance
  – Not the overall system performance
  – Left out CPI_{base} in the calculation
C5 CPI

- CPI calculation used for overall system performance comparison, such as speedup of computer A to computer B
- \( CPI = CPI_{\text{base}} + \text{Penalty} \)
  - \( CPI_{\text{base}} \) is the CPI without the special case penalty.
  - Penalty is the penalty cycle per instruction
C5 CPI Example

• CPI Calculation Example:
  CPI for two leveled cache. Assume unified L2 and separate instruction and data cache.

\[
\text{CPI} = \text{CPI}_{\text{base}} + \text{Penalty}
\]

\[
\text{CPI}_{\text{base}} \text{ depends on the program and processor.}
\]

\[
\text{Penalty} = \text{L1 miss penalty} + \text{L2 miss penalty}
\]

\[
\text{L1 miss penalty} = \text{data L1 miss penalty} + \text{instruction miss penalty}
\]

\[
\text{Data L1 miss penalty} = \text{data L1 access per instr} \times \text{data L1 miss rate} \times \text{data L1 miss penalty.}
\]

\[
\text{Instruction miss penalty} = \text{instruction L1 access per instr} \times \text{instruction L1 miss rate} \times \text{instruction L1 miss penalty.}
\]

\[
\text{L2 miss penalty} = \text{L2 access per instr} \times \text{L2 miss rate} \times \text{L2 miss penalty.}
\]

\[
\text{L2 access per instr} = \text{instruction L1 access per instr} \times \text{instruction L1 miss rate} \times \text{data L1 access per instr} \times \text{data L1 miss rate}
\]
Easier to program in virtual memory
Additional hardware needed to translate between the two
OS is usually used to translate the two
TLB is used to cached the translation result for faster translation.
C5 VM

- CPI calculation for memory with VM

\[ CPI = CPI_{\text{base}} + \text{Penalty} \]

Penalty = TLB miss penalty cycle per instruction

\[ = \text{TLB miss per instruction} \times \text{penalty cycle per TLB miss} \]

\[ = \text{TLB access per instruction} \times \text{TLB miss rate} \times \text{penalty cycle per TLB miss} \]

TLB access per instruction is for both data and instruction access.
C7 Disk

• Average disk access time

  = average seek time + average rotational delay
    + transfer time + controller overhead

• Average rotational delay = time for platter to
  rotate half a cycle

• Transfer time = size of access / transfer speed

• Transfer speed = rotational speed * size of tracks
  – Assuming the bit are read off continuously as the disk
    head pass over it.
C7 RAID

• Use small disks to build a large storage system
  – Smaller disks are mass produce and so cheaper
  – Large number of disks results high failure rate
  – Use redundancy to lower failure rate

• RAID 2: Mirror
• RAID 3: bit interleave
• RAID 4/5 : distributed bit interleaving
C7 RAID

• Mean time to data loss

\[ \text{MTTDL} = \frac{\text{MTTF}_{\text{disk}}^2}{N \times (G-1) \times \text{MTTR}_{\text{disk}}} \]

- \( N \) = total number of disks in the system
- \( G \) = number of disks in the bit protected group
- \( \text{MTTR} = \text{mean time to repair} \)
  \[ = \text{mean time to detection} + \text{mean time to replacement} \]

\[ 1/\text{MTTF} = \sum_{\text{All component}} 1/\text{MTTF}(\text{component}) \]
C7 RAID

- **MTTDL example**
  RAID 2 system with 10 GB total capacity. Individual disk are 1 GB each with MTTF_{disk} 1000000 hr. Assume MTTR of 1 hr

- **Solution**
  - G = 2 since each disk is mirrored.
  - N = 20, 10 for 10 GB of capacity 10 for mirroring
  - \[ \text{MTTDL} = \frac{\text{MTTF}_{\text{disk}}^2}{(N*(G-1)*\text{MTTR}_{\text{disk}})} \]
  - \[ = 1000000^2 / (20*(2-1)*1_{\text{disk}}) \]
C7 Queuing Theory

• Used to analyze system resource requirement and performance
  + more accurate than the simple extreme case study
  + less complicated than the full simulation study

• Results are based on exponential distribution modeling of the arrival rate
C8

- Class of connections ordered by decreasing distance, bandwidth, latency
  - Wan/internet
  - LAN
  - SAN
  - Bus
• **Total transfer time**

  \[ \text{Total transfer time} = \text{sender overhead} + \text{Time of flight} + \frac{\text{message size}}{\text{bandwidth}} + \text{receiver overhead} - \text{Latency} \]

  - *Latency* = sender overhead + Time of flight + receiver overhead
  - Hard to improve latency, easy to improve bandwidth

• **Effect bandwidth**

  \[ \text{Effect bandwidth} = \frac{\text{Message size}}{\text{Total transfer time}} \]

  - Total transfer time is dominated by latency as bandwidth increases.
  - Need bigger message size to ameliorate the latency overhead
C6 Multiprocessor Limit

• Amdahl’s Law
  – Speedup
    \[ = \frac{1}{(\text{fraction enhanced/speedup}) + \text{fraction not enhanced}} \]

• Fraction enhanced is limited by sequential portion of the program

• Communication overhead may limit speedup for the parallel portion.
C6 scaling calculation for app

• Matrix multiplication. Take the squaring special case. A*A
  – A is a square matrix with n elements.
  – P is the number of processors

Computation scaling:
  A is $n^{0.5} \times n^{0.5}$ matrix. Calculation complexity is then $n^{1.5}$
  Diving between P processor gives $(n^{1.5})/P$

Communication scaling:
  Assume matrix A is tiled into square with side dimension $n^{0.5}/P^{0.5}$
  elements needed are row and column this gives
  $2 \times [(n^{0.5}/P^{0.5})(n^{0.5}) - (n^{0.5}/P^{0.5})] \sim n/P^{0.5}$

Computation to communication scaling
  $n^{0.5}/P^{0.5}$
C6 Type of multiprocessor

- **SISD**: single instruction single data
  - 5 stage pipelined RISC processor
- **SIMD**: single instruction multiple data
  - Vector processor
- **MISD**: multiple instruction single data
- **MIMD**: multiple instruction multiple data
  - Super scalar, clustered machines, VLIW
C6 dealing with parallelism

• Shared memory
  – Communication between processors are implicit

• Message passing
  – Explicit communication

• They are equivalent in term of functionality. Can build on from another