

CHALLENGES OF PARALLEL PROCESSING

INSUFFICIENT PARALLELISM

Example 1:

Suppose you want to achieve a speedup of 80 with 100 processors. What fraction of the original computation can be sequential?

Answer Recall from [Chapter 1](#) that Amdahl's law is

$$\text{Speedup} = \frac{1}{\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} + (1 - \text{Fraction}_{\text{enhanced}})}$$

For simplicity in this example, assume that the program operates in only two modes: parallel with all processors fully used, which is the enhanced mode, or serial with only one processor in use. With this simplification, the speedup in enhanced mode is simply the number of processors, while the fraction of enhanced mode is the time spent in parallel mode. Substituting into the previous equation:

$$80 = \frac{1}{\frac{\text{Fraction}_{\text{parallel}}}{100} + (1 - \text{Fraction}_{\text{parallel}})}$$

Simplifying this equation yields:

$$0.8 \times \text{Fraction}_{\text{parallel}} + 80 \times (1 - \text{Fraction}_{\text{parallel}}) = 1$$

$$80 - 79.2 \times \text{Fraction}_{\text{parallel}} = 1$$

$$\text{Fraction}_{\text{parallel}} = \frac{80 - 1}{79.2}$$

$$\text{Fraction}_{\text{parallel}} = 0.9975$$

Thus, to achieve a speedup of 80 with 100 processors, only 0.25% of the original computation can be sequential. Of course, to achieve linear speedup (speedup of n with n processors), the entire program must usually be parallel with no serial portions. In practice, programs do not just operate in fully parallel or sequential mode, but often use less than the full complement of the processors when running in parallel mode.

LONG LATENCY REMOTE COMMUNICATION

Example 2:

Suppose we have an application running on a 32-processor multiprocessor, which has a 200 ns time to handle reference to a remote memory. For this application, assume that all the references except those involving communication hit in the local memory hierarchy, which is slightly optimistic. Processors are stalled on a remote request, and the processor clock rate is 3.3 GHz. If the base CPI (assuming that all references hit in the cache) is 0.5, how much faster is the multiprocessor if there is no communication versus if 0.2% of the instructions involve a remote communication reference?

Answer It is simpler to first calculate the clock cycles per instruction. The effective CPI for the multiprocessor with 0.2% remote references is

$$\begin{aligned}\text{CPI} &= \text{Base CPI} + \text{Remote request rate} \times \text{Remote request cost} \\ &= 0.5 + 0.2\% \times \text{Remote request cost}\end{aligned}$$

The remote request cost is

$$\frac{\text{Remote access cost}}{\text{Cycle time}} = \frac{200 \text{ ns}}{0.3 \text{ ns}} = 666 \text{ cycles}$$

Hence, we can compute the CPI:

$$\text{CPI} = 0.5 + 1.2 = 1.7$$

The multiprocessor with all local references is $1.7/0.5 = 3.4$ times faster. In practice, the performance analysis is much more complex, since some fraction of the noncommunication references will miss in the local hierarchy and the remote access time does not have a single constant value. For example, the cost of a remote reference could be quite a bit worse, since contention caused by many references trying to use the global interconnect can lead to increased delays.

PERFORMANCE OF SYMMETRIC SHARED-MEMORY MULTIPROCESSORS

Example 3:

Assume that words x1 and x2 are in the same cache block, which is in the shared state in the caches of both P1 and P2. Assuming the following sequence of events, identify each miss as a true sharing miss, a false sharing miss, or a hit.

Any miss that would occur if the block size were one word is designated a true sharing miss.

Time	P1	P2
1	Write x1	
2		Read x2
3	Write x1	
4		Write x2
5	Read x2	

Answer Here are the classifications by time step:

1. This event is a true sharing miss, since x1 was read by P2 and needs to be invalidated from P2.
2. This event is a false sharing miss, since x2 was invalidated by the write of x1 in P1, but that value of x1 is not used in P2.
3. This event is a false sharing miss, since the block containing x1 is marked shared due to the read in P2, but P2 did not read x1. The cache block containing x1 will be in the shared state after the read by P2; a write miss is required to obtain exclusive access to the block. In some protocols this will be handled as an *upgrade request*, which generates a bus invalidate, but does not transfer the cache block.
4. This event is a false sharing miss for the same reason as step 3.
5. This event is a true sharing miss, since the value being read was written by P2.

MODELS OF MEMORY CONSISTENCY: AN INTRODUCTION

Example 4:

Suppose we have a processor where a write miss takes 50 cycles to establish ownership, 10 cycles to issue each invalidate after ownership is established, and 80 cycles for an invalidate to complete and be acknowledged once it is issued. Assuming that four other processors share a cache block, how long does a write miss stall the writing processor if the processor is sequentially consistent? Assume that the invalidates must be explicitly acknowledged before the coherence controller knows they are completed. Suppose we could continue executing after obtaining ownership for the write miss without waiting for the invalidates; how long would the write take?

Answer When we wait for invalidates, each write takes the sum of the ownership time plus the time to complete the invalidates. Since the invalidates can overlap, we need only worry about the last one, which starts $10 + 10 + 10 + 10 = 40$ cycles after ownership is established. Hence, the total time for the write is $50 + 40 + 80 = 170$ cycles. In comparison, the ownership time is only 50 cycles. With appropriate write buffer implementations, it is even possible to continue before ownership is established.

INCLUSION AND ITS IMPLEMENTATION

Example 5:

Assume that L2 has a block size four times that of L1. Show how a miss for an address that causes a replacement in L1 and L2 can lead to violation of the inclusion property.

Answer Assume that L1 and L2 are direct mapped and that the block size of L1 is b bytes and the block size of L2 is $4b$ bytes. Suppose L1 contains two blocks with starting

addresses x and $x + b$ and that $x \bmod 4b = 0$, meaning that x also is the starting address of a block in L2; then that single block in L2 contains the L1 blocks x , $x + b$, $x + 2b$, and $x + 3b$. Suppose the processor generates a reference to block y that maps to the block containing x in both caches and hence misses. Since L2 missed, it fetches $4b$ bytes and replaces the block containing x , $x + b$, $x + 2b$, and $x + 3b$, while L1 takes b bytes and replaces the block containing x . Since L1 still contains $x + b$, but L2 does not, the inclusion property no longer holds.