NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

TDT4200: Problem set 1

Magnus Moan

September 18, 2017

1 THEORY

1.1 CACHING

A. All three are policies for placing and accessing memory blocks in the cache. First some general terminology regarding cache memory. The cache consists of a number of slots. The number of slots is always a power of 2. A cache slot consists of cache lines. The number of cache lines is usually a power of 2 as well.

Fully associative is a policy where a block of memory can be placed in any cache slot. In practice the cache is iterated until one finds the memory block one is looking after. The search is conducted by comparing the tag of the memory block one is looking after with the tags of the cache slots. The tag of the memory block is a number which is part of the memory address of the memory block one is looking after. Exactly which part of the memory address that makes up the tag depends on the size of the address (32 vs. 64 bits) and the number of bytes in each cache line.

Direct mapped is a policy where a block of memory is placed in a specific cache slot given by the memory blocks address. This is best demonstrated by an example: Considered a cache with 128 slots and 32 bytes per slot. Say we are working with 32 bit memory addresses. In this case the cache slot number of a specific memory block is given as the number represented by bits 5-11 in the memory address of the block. Bits 0-4 are the offset within the memory slot. This comes from the fact that we need 5 bits to represent the 32 possible bytes for each slot and we need 7 bytes to represent the 128 different slots.

n-way set associative is a policy that combines fully associative caching and direct mapped caching. In a n-way set associative caching the cache is divided into n sets. Each

memory block is placed into the cache by placing it into a specific set. Within the set the memory block is placed in the same fashion as in fully associative caching, at the first available spot. Which set to place a specific memory block in is determined by the memory address of the given memory block in the same fashion as in direct mapped caching.

B. Pacheco emphasizes that the cache is controlled by hardware and it's possible for a programmer to directly influence which data that is in the cache at what time. However, he also states that if we where to know what cache policy the underlying hardware is using we could indirectly have some control over caching. Pacheco exemplifies this using two dimensional arrays in the programming language C. Two-dimensional arrays are in practice stored as one long one-dimensional array. If we were to loop over such an array it would be important to loop over it in such a fashion that would generate as few misses as possible when trying to access the cache. One must keep in mind which part of the array that gets added to the cache when one tries to access a value at an specific index. By keeping this in mind one could potentially generate a lot fewer misses compared to iterating over the arrays in a fashion that forces memory to be added to the cache more times than necessary.

C. When operating with multiple caches working on the same shared memory one could experience cache incoherence if one of the caches where to update the memory without notifying the other. The two main mechanisms for dealing with this issue are:

Snooping is a cache coherence ensurer which works by monitoring the parts of the memory it has cached. If another cache where to access such parts of the memory the and in turn write to such a memory location the snooping mechanism will invalidate that memory in its own cache.

Directory-based is a cache coherence mechanism which works by maintaining a common directory which ensures coherence between caches.

1.2 Gustafson's law

A. In the case where $f = 0$ the program fits Gustafson's Law. The reason for this is that the linear part of the program does not scale with the problem size since the linear part will always only be run once, no matter of large problem_size gets. The opposite argument means that if $f = 1$ the program fits Amdahl's Law.

B. Amdahl's Law is defined as:

$$S_l(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

Gustafson's Law is defined as:

$$S_l(s) = 1 - p + sp$$

here we have that S_l is the performance limit, p is the fraction of program that is parallellized and s is the speed up of the part that is parallellized. The performance limits are given in the table below.

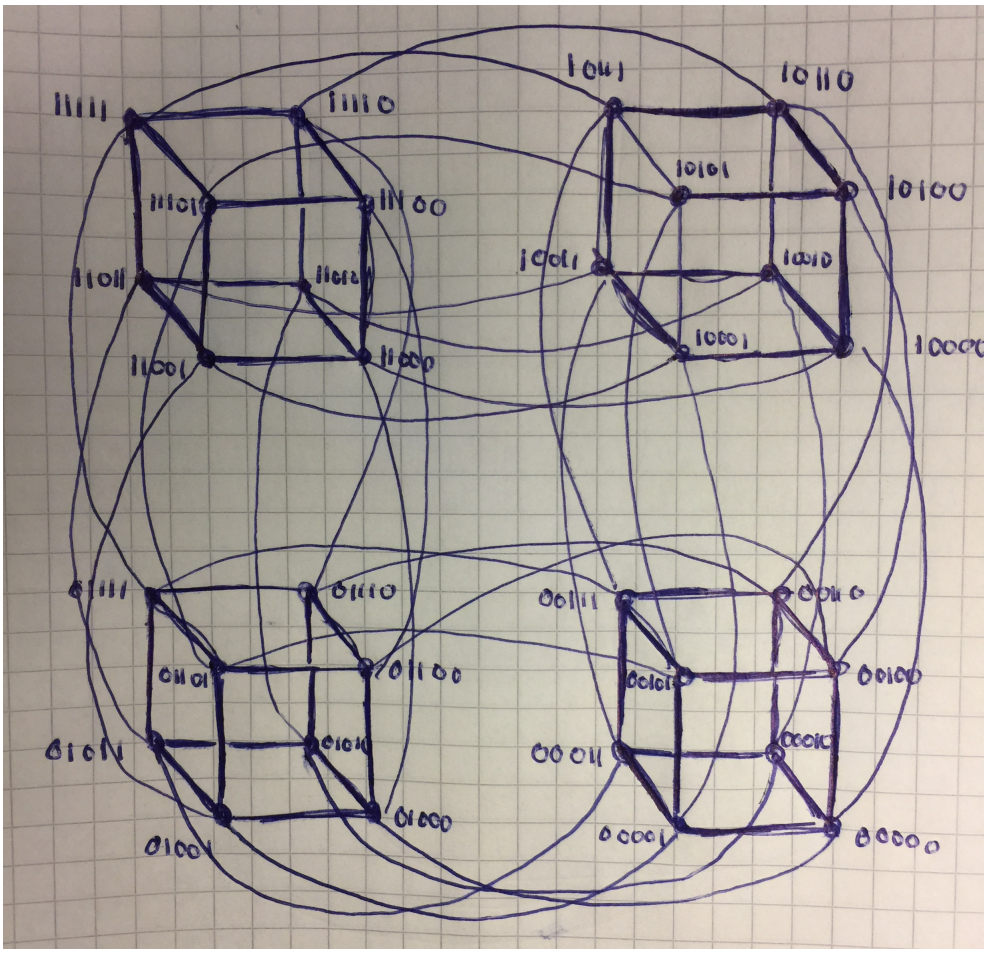
Table 1.1: Theoretical speedups

	Amdahl			Gustafson		
Proc.	2	4	8	2	4	8
$f = 0$	1.82	3.10	4.71	1.9	3.7	7.3
$f = 1$	1.33	1.60	1.78	1.5	2.5	4.5

C. The two laws give different results because they do different assumptions. Amdahl's law operates with a fixed data size, which Gustafson's law don't. This leads to the big difference in the results from the two laws.

1.3 Connection Networks

A.



B.