PS3 Theory (TDT4200)

Stian Jensen

Problem 1, MPI Derived types

a) Explain the purpose of derived types in MPI

MPI based applications may need to transfers many different types of data, more complicated than simple INTs or FLOATs. Derived data types allows the programmer to define his own datatypes with custom offsets between blocks, and custom sizes.

b)
Array b:

			x						
			x						
			x						
			x						
			x						
			x						
			x						
			x						
			x						
			x						

Array c:

	x	x							
		x	x						
			x	x					
				x	x				
					x	x			

Problem 2, Memory & Caching

a) Three types of cache misses

- **Compulsory:**
  - The first time you access a block, it will have to be loaded into the cache.
- **Capacity:**
  - Occurs when there isn't room in the cache for all the elements you want to access. When the cache is full and something new is added, a block will have to be discarded.
- **Conflict:**
  - Two different blocks are mapped to the same location. They cannot both be stored at the same time.

b) What is temporal and spacial locality?

Temporal locality means that if at one point in time, a block is accessed, then it is likely that it will soon be accessed again.

Spacial locality means that when a block is accessed it is likely that nearby memory locations will be accessed soon.

c)

I) Each variable is only accessed once, so I would not say this exhibits temporal locality. A jump of 1000 between each index, means there's not really any spacial locality, either.
II) Here too, each variable is only accessed once, thus no temporal locality. However, here the index is only incremented by one in each run of the loop, which means that this is an example of spacial locality.
III) Here, in the double nested loop, the inner loop actually executes the same actions 100 times. This is therefore an example of temporal locality. As in the first example, though, no spacial locality.

Problem 4, Branching

a) What is branch prediction? How can it improve performance?

Branch prediction attempts to guess which branch will be taken before it is known for sure.

b)

The formula for execution time for each iteration of the loop with a perfect branch predictor:

$r*f + (1-r) * s + p$

I)
The formula for the execution time, when the branch predictor always predicts the slow branch:

$r*f + (1-r) * s + p + r*b$

We then get that when r is as in the following expression, it will be beneficial to always use the slow function:

$r < - p / (f-s+b)$

II)

Execution time on average will be:

$r*f + (1-r) * s + p + b$

It is beneficial to always use the slow function when:

$r < (b+p) / (s-f)$

Problem 5, Optimization

a) Explain why this optimization in some cases can increase and in others decrease, performance

Some times the amount of different possible output from the function can be very large, which means a large amount of memory is consumed for storing all the data.

b)

Assuming each integer to be 32 bit, there is room for 16 integers in each cache line.

Without a table the total execution time will be: 128 * 4 * 1024 * 5 * h = 2621440 * h
With table but no cache: 128 * 4* 1024 * 10 * h = 5242880 * h
With table and 1 cache line (assuming a whole cache line is filled when data is read from memory):
128 * 4 * 64 * (10 * h + 15 * h) = 819200 * h

We see that already by introducing one 64 byte cache line, the execution time is much improved.