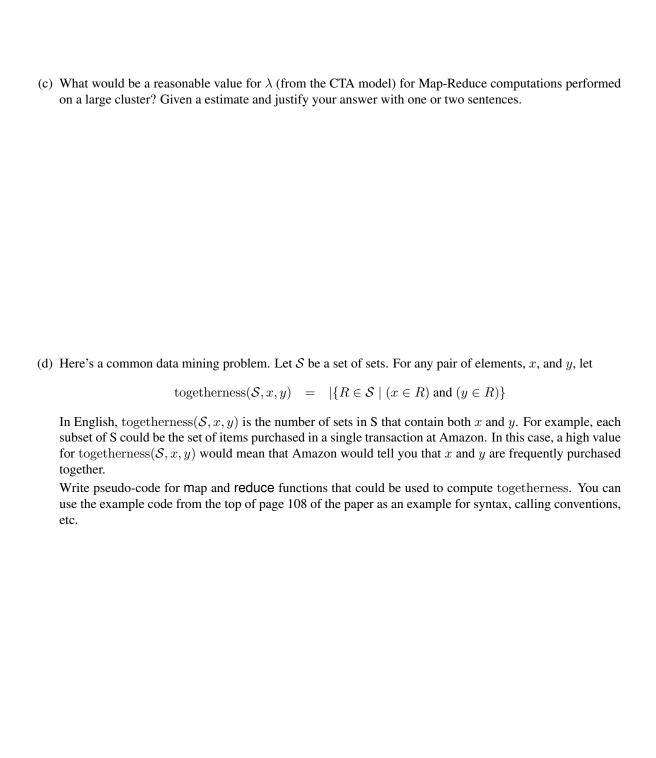
1. (20 points) Give a one or two sentence definition for each term or phrase below (5 points each).	
(a) False sharing.	
(b) Simultaneous multi-threading.	
(c) One-sided communication.	
(+)	
(d) Toil requiries	
(d) Tail recursion.	

<ol> <li>(40 points) The following questions pertain to the paper, MapReduce: Simplified Data Processing on Large Clusters. Each of these questions has a value of 10 points.</li> </ol>
(a) The paper describes a Map-Reduce paradigm. Would it be practical to extend this to Scan-Reduce? Why or why not?
(b) Briefly describe what happens if a machine taking part in a Map-Reduce computation crashes before the computation is complete. Describe a Map-Reduce computation where the result is the same whether or not some machine crashes. Describe a computation where a different result can be produced because a machine crashes. Your answer to all three of these questions about machine crashes should take <i>at most</i> eight sentences.



3.	(30 r	<b>points</b> ) The following questions pertain to the paper, <i>The GPU Computing Era</i> . Each of these questions has
		lue of 10 points.
	(a)	Assume that each CUDA core can perform one double-precision, floating-point operation every 4ns. What is the peak-computation rate for the Fermi GPU shown in Figure 3?
	(b)	A double-precision floating point number is eight bytes. Assume that each double-precision operation hat two operands. If each value stored on a Fermi GPUs caches and registers is used once, how long does take for the GPU to consume all of its on chip data?
	(c)	A Fermi GPU can read data from off-chip DRAM at a peak rate of 144GB/sec. If a Fermi GPU is to operate at its peak-computation rate, how many double-precision operations must it perform for each double-precision value read from memory? For simplicity, you can pretend that there is nothing to be written back to memory.

- 4. (10 points) Do any one (but not more) of the following three questions:
  - (a) Sketch a way to compute matrix-multiplication using Map-Reduce.
  - (b) Sketch a way to compute count-3s using CUDA.
  - (c) Given an array x[i] for  $0 \le i < n$ , use generalized reduce or scan to compute y[i] with

$$y[i] = x[0]^{x[1]^{x[2]} \cdot x[i]}$$