EEE4084F Quiz 2 2014 SOLUTONS
Syllabus: Lectures 5-7 + Seminar 3

Q1 a) Cache coherency basically means that the memory caches between different processors are kept consistant. i.e. writing a value to an address X ensures that the new value is reflected in the caches corresponding to that memory element for the other processors. Cache coherency is relevant more for a shared memory system with separate cache for each processor. When one variable is changed, the other copies of the same variable must be changed also in the cache of the other processors (if they are accessing the same variable).

Q1 b) SMP = Symmetric Multi-Processors
The memory accesses time is the same for all the processors (synchronous memory access timing)

Q1 c) Advantages:
- All processors have access to all memory in a global address space.
- Global address space gives a user-friendly programming approach
- Sharing data between tasks is fast and uniform due to the proximity of memory to CPUs
- Processors operate independently, but can share same global memory.
- Changes to global mem by one processor immediately seen by others.

Disadvantages:
- Major drawback: lack of scalability between memory and CPUs.
- Adding CPUs can increases traffic on shared memory-CPU path (for cache coherent systems also increases traffic associated with cache/memory management)

Q2 a) GPU = Single Instruction Multiple Data (SIMD) [2 marks]
Basically you want to explain the gyst of SIMD. Perhaps a little consoling naration to Mr Jobs…
The algorithms you should consider for a GPU are more likely the embarasingly parallel sort; coarse-gained problems that can be easily distributed to a whole lot of processing cores, but where each operation for a bank of processors can be done in-step. You also want operations that either don't need to transfer much results between each other (i.e. storing results in memory for later transfer back to the CPU) -- you want to work in blocks of computation, where a whole bank of intermediate results are generated simultaneosuly, preferably without any synchrnonization/mutexes needed, then saved to memory on the GPU and these intermediate results pass on to another block of processing operations that then starts and continues working on these results until the final results are genrated for the CPU to read back.

Q2 b) This sounds like a replicate & reduce or master/slave situation -- but more likely replicate & reduce because there are supposed to be two multi-core machines.

{

We could have a short contemplation of the processing architecture before committing to a memory architecure....

Consider the replicate & reduce approach, the master would run on core 1 of machine 1 obtaining the needed data. Since there are four applications, these applications could run on the two machines. Perhaps another core on the main machine would be needed to replicate data over to the second machine - but this could more efficiently just be cone via DMA to the network controller by the main core (instead of using IPC) so that means another two cores for each application... that is looking like machine 1 needs to be a quad core. The other machine is running another two programs, if these programs are essentially sequential, then a core each. This suggest machine 2 could just be a dual core machine (like a Core2 Duo) which could provide some cost saving. Both would have GPUs that could assist the CPU cores in doing parallel processing.

}

So, in terms of memory architecutre I would suggest a hybrid system. Since there are two computers involved, a shared memory approach alone isn't going to solve the problem -- so it would need some distributed memory so that the main processor generating data can share the data with the other machine, and also local shared memory on each machine so that this data can be accessed and worked on my different threads, possibly threads running in different programms.

Q3 a) Bisection Bandwidth: A measure of how much data flows from one half of the processor to the other if the system is figuratively “bisected.”. This means that for a system that is segmented into two equal parts, the bandwidth between the two parts is called the bisection bandwidth.

Q3 b) 1. Processing Objectives: Backend - Refinement of information and target parameters received from front-end: target tracking, etc.; while Front-end - Data Capturing:Remove noise etc. 2. Flow of Control: Backend - Complex flow of control; Front-end - Linear flow of control. 3. Data Types: Back-end - Composite data-structure (objects); Front-end - Signals and Images. 4. Input: Back-end - Features, detections, state vectors, and images; Front-end -Single-precision fixed-point continuous or regular streams. 5. Program Complexity: Backend - Around 20,000 to 50,000 SLOC; Frontend - Greater than 100,000 SLOC.

Q4 a) 1. Understand the problem
2. Partitioning (separation into main tasks)
3. Decomposition & Granularity.

Q4 b) Partitioning, decomposition, and communications.

Q4 c) Domain decomposition - data associated with a problem is decomposed and each parallel task then works on a portion of that data, while functional decomposition focuses on the computation that is to be performed rather than on the data manipulated by the computation. The problem is decomposed according tasks (Work to be done). Each task then performs a portion of the overall work.

Q5 a) Synchronous communication - Handshaking between tasks that are sharing data required and Referred to as blocking communications since other work must wait until the communications have completed while Asynchronous communication - Tasks to transfer data independently from one another and non-blocking.

Q5 b) LVDC (Launch Vehicle Digital Computer) – Appolo Saturn V mission computer.

Q6 256 bytes