Q1:

1.The statement is false. Since the static priorities of A and B are equal, the only thing that differentiate them is their bonus. Since bonus varies only according to sleep_average, and B is IO-bound while A is CPU-bound, the sleep_average of B is bigger than the sleep_average of A, therefore the bonus of B is higher than A's. Which means that the prio of B is lower than that of A (since prio = static_prio - bonus).

2. A will get a bigger timeslice than B and will be executed earlier than B in epoch (since the processes are arranged according to their priorities in the runqueue).

3. A will get a bigger timeslice than B. A will be executed earlier than B in epoch. More than that A will get a better reaction time A can stop executing of B when returning from waiting if it still has remaining timeslice.

4.

a. Yes. Let's take a look at all possible cases:
- Current process is the only one that has curr->prio priority and it's the highest priority in the run queue.
- Current process isn't the only one that has curr->prio priority and it's the highest priority in the run queue
- Current process is the only one that has curr->prio priority and there is a process in active with higher priority

Q2:

1. The difference between working with multiple CPUs and a single CPU, is that when working with multiple CPU, a CPU's schedule function checks wether it's run queue is empty or not. If it is, load_balance is called.load_balance looks for the busiest run queue and moves tasks from it to the idle CPU if the busiest run queue is significantly busy.
2. pros: If each CPU has it's own run queue, each CPU can access it's run queue quickly and efficiently, without worrying about receiving data contamination from other CPUs or causing data contamination to other CPUs. cons: If each CPU has its own run queue, its possible that 1 run queue will have multiple processes in the ready mode while another CPU runs a process from its own run queue with a lesser priority. In such cases we see that some CPUs might not be working on the highest priority process that is technically available.
3. a scheduling domain is a hierarchical system that separates CPUs into different groups. It helps speed up the balancing function by allowing a CPU to find a busy CPU that needs to get some load off without iterating through all the CPUs in the computer.
4. Once every tick, reschedule_tick is invoked, it iterates over the path from the local domain to the top level domain and checks if any of the domains need rebalancing. The algorithm has statistical information that represents the current average load in a CPU and in a group. and if it sees 2 CPUs and 2 groups who's total average load can be lessened by moving a few processes from 1 CPU to another, it decides to rebalance.