CHAPTER 20

Database-System Architectures

Practice Exercises

20.1 Is a multiuser system necessarily a parallel system? Why or why not?

20.2 Atomic instructions such as compare-and-swap and test-and-set also execute a memory fence as part of the instruction on many architectures. Explain what is the motivation for executing the memory fence, from the viewpoint of data in shared memory that is protected by a mutex implemented by the atomic instruction. Also explain what a process should do before releasing a mutex.

20.3 Instead of storing shared structures in shared memory, an alternative architecture would be to store them in the local memory of a special process and access the shared data by interprocess communication with the process. What would be the drawback of such an architecture?

20.4 Explain the distinction between a *latch* and a *lock* as used for transactional concurrency control.

20.5 Suppose a transaction is written in C with embedded SQL, and about 80 percent of the time is spent in the SQL code, with the remaining 20 percent spent in C code. How much speedup can one hope to attain if parallelism is used only for the SQL code? Explain.

20.6 Consider a pair of processes in a shared memory system such that process *A* updates a data structure, and then sets a flag to indicate that the update is completed. Process *B* monitors the flag, and starts processing the data structure only after it finds the flag is set.

Explain the problems that could arise in a memory architecture where writes may be reordered, and explain how the sfence and lfence instructions can be used to ensure the problem does not occur.

20.7 In a shared-memory architecture, why might the time to access a memory location vary depending on the memory location being accessed?

20.8 Most operating systems for parallel machines (i) allocate memory in a local memory area when a process requests memory, and (ii) avoid moving a process from one core to another. Why are these optimizations important with a NUMA architecture?

20.9 Some database operations such as joins can see a significant difference in speed when data (e.g., one of the relations involved in a join) fits in memory as compared to the situation where the data do not fit in memory. Show how this fact can explain the phenomenon of **superlinear speedup**, where an application sees a speedup greater than the amount of resources allocated to it.

20.10 What is the key distinction between homogeneous and federated distributed database systems?

20.11 Why might a client choose to subscribe only to the basic infrastructure-as-a-service model rather than to the services offered by other cloud service models?

20.12 Why do cloud-computing services support traditional database systems best by using a virtual machine, instead of running directly on the service provider's actual machine, assuming that data is on external storage?