Question 1
What is a parallel computer? Can a computer with only one processing core be considered a parallel computer? Explain.

Answer: (from lecture): A parallel computer is a computer system that uses multiple processing elements simultaneously in a cooperative manner to solve a computational problem.

Yes, a single processing core can still provide parallelism, e.g., instruction-level parallelism (through pipelining hardware) and data parallelism (through vector registers and instructions).

Question 2
What are the three components of parallelism?

Answer: concurrency, parallel hardware, and performance

Question 3
What is the difference between a shared memory parallel system and a distributed memory parallel system? Give two advantages for each?

Answer: The main difference is that a shared memory parallel system provides a single physical memory address space at the hardware level, such that the same memory address used by two processors will reference the same physical memory location.

Shared memory advantages:

- Easier to program because it looks more like the sequential programming model.
- Can achieve higher performance and be more efficient at small scale parallelism.
- Supports multithreading and enables faster synchronization between threads of execution.
- Easier to support a single image OS.

Distributed memory advantages:

- Easier to scale to large parallelism. (In fact, it is the only way to scale to very large machine sizes.)
- Not as costly to scale, since you can use commodity components and you don’t have to worry about scaling the memory system.
- Optimizing parallel programs can be easier in the absence of memory sharing effects.
- Can achieve higher performance at large-scale parallelism.
Question 4
What is a NUMA parallel architecture and why was it invented? Give at least one reason.

**Answer:** A NUMA parallel architecture is one where memory access (MA) is non-uniform (NU) in that different addressable memory locations have a different (i.e., not the same) access latency. The architecture was developed to help address the difficulties (mostly, with respect to architecture scaling) encountered with maintaining uniform memory access latencies in shared memory parallel systems. It provides a less costly architecture while facilitating the use of hierarchical memory systems, including caches, in modern processor architectures.

Question 5
Why do we have multi-core processors? Are not single-core processors good enough? Why is multi-core a disruptive technology from the point of view of parallel computing?

**Answer:** The need for multicore is based on the economics by which computer vendors make money (they have to produce "faster") chips regularly and simple physics limitations having to do with increased heat dissipation due to increasing dynamic power consumed being proportional to CPU frequency, so simply scaling up the frequency is not sustainable (Dennard scaling discussed in lecture).

“Disruptive” from the point of parallel computing means that existing algorithm design approaches and most software will not work on these architectures without being significantly changed or redone.

Question 6
For large-scale parallel systems, the interconnection network is key. Would you agree? Explain why or why not.

**Answer:** Yes, because a slow network means that the same algorithm would have much worse parallel efficiency than it would on a faster network for any computation that is not embarrassingly parallel, which would make solving larger problems harder or impossible. Not having a network at all would make it impossible to compute anything but a small handful of problems in parallel.

Question 7
What distinguishes Amdahl’s Law from Gustafson-Baris’ Law with respect to parallel speedup?

**Answer:** Amdahl’s Law defines parallel speedup with respect to the sequential execution time of a fixed-size problem. It represents this sequential time with respect to the part that can be parallelized and the part that remains sequential. In contrast, Gustafson-Baris’ Law defines parallel speedup with respect to the parallel execution time, which can increase depending on the size of the problem being computed. As the problem size grows, the parallel fraction incorporates more work, thus allowing greater speedups to be achieved.

Question 8
For a given problem size, why does the efficiency go down as the number of processing elements increase? Is this always true? If not, give an example.
Answer: Efficiency generally goes down because overhead of parallel computation increases with more processing elements. It is not true that efficiency decreases because there may be some execution effects that occur during sequential execution that do not occur in the same way in parallel execution, leading to improved performance that offsets extra parallel overhead. For example, going from one core to two doubles the amount of cache available and increases the available bandwidth to memory, which may lead to improvements by more than a factor of 2 and hence efficiency is more than 100% (superlinear speedup).

Question 9

Suppose you are comparing two algorithms, A and B, for the same problem. Suppose that algorithm A has better strong scaling than algorithm B on a parallel machine.

a) Will algorithm A always have better weak scaling than algorithm B on this machine? Explain.

Answer: First, let us rephrase what we are given. Algorithm A has better strong scaling than B, by Amdahl’s law translates to $S_A > S_B$ where $S_A$ is the fixed-problem-size speedup for A and $S_B$ is the fixed-problem-size speedup for B, computed in terms of the serial fractions $f_A$ and $f_B$ (lecture 3, slide 12):

$$S_A > S_B \rightarrow \frac{1}{f_A + (1 - f_A)/p} > \frac{1}{f_B + (1 - f_B)/p} \rightarrow \frac{f_A + (1 - f_B)/p}{f_B + (1 - f_A)/p} > 1 \rightarrow \frac{(p-1)f_B + 1}{(p-1)f_A + 1} > 1 \rightarrow (p-1)f_B + 1 > (p-1)f_A + 1 \rightarrow f_B > f_A.$$

Given this, can we show that the weak scaling as defined by Gustafson-Barsis for algorithm A is always better than the weak scaling for B, i.e., can we derive the following inequality given the sequential fraction relationships $f_B > f_A$, is it always true that $weakS_A > weakS_B$?

Recall that the parallel time in weak scaling is fixed, so $T_p^A$ and $T_p^B$ are constants. If we assume that the sequential portions of $T_p$ are constant (i.e., $f_A$ and $f_B$ do not change as we increase the number of processors), then for any $p > 1$, weak scaling of A is always better than B:

$$f_B > f_A \rightarrow 1 + (p-1)(1 - f_A) > 1 + (p-1)(1 - f_B) \rightarrow weakS_A > weakS_B$$

However, if the sequential portion of $T_p$ for each algorithm does not remain constant as more processors are used and the problem size increases (e.g., the sequential fraction of algorithm A may have complexity $O(n^2)$, while the sequential fraction of algorithm B may have complexity $O(n)$), then for very large $p$, algorithm B will have better weak scaling than A.

b) Is it possible that algorithm B will have better strong scaling than algorithm A on a different parallel machine?

Answer: Yes, we did not specify anything about these algorithms, so it’s entirely possible for the sequential fraction of A to be larger on a different architecture ($f_A > f_B$).

c) Amdahl’s Law is defined with respect to the fastest sequential execution time. Suppose the fastest sequential algorithm on machine X, $S_x$, is not the fastest sequential algorithm on machine Y, $S_y$. Does this cause a problem when comparing speedups between the two machines? Explain.

Answer: No, speedup depends on the non-parallelizable, sequential, fraction of the algorithm, which is defined in a machine-independent manner. Speed up as defined by Amdahl’s Law is not dependent on the actual total sequential execution time on any particular machine.
Problem 10

Consider the problem of computing the dot product of two vectors, $A$ and $B$, each of length $N$. The dot product is defined as:

$$A \cdot B = \sum_{i=1}^{N}(A_iB_i)$$

a) Describe how you would parallelize this problem.

**Answer:** First, the $A_i \cdot B_i$ operations can be done entirely in parallel. Given $N$ elements, there is $N$-way parallelism. For $P \leq N$, give each of $P$ processors $N/P$ $A$ and $B$ elements to compute the products without any communication.

Second, we need to sum the products using a binary tree reduction.

b) Assume that multiplying two numbers takes 4 units of time, adding two numbers takes 2 units of time, and communicating one number between two processing elements takes 50 units of time. What is the parallel runtime, speedup, and efficiency of your parallel algorithm when run on $p$ processing elements? You can assume $n$ and $p$ are powers of 2. If you can, write your answer in terms of computation time and communication time components. [10pts]

**Answer:**

- **Sequential runtime** = $4 \times N + 2 \times (N - 1) = 6 \times N - 2$
- **Parallel runtime** = $4 \times N/P + 2 \times (N/P - 1) + 2 \times \log P + 50 \times \log P = 6 \times N/P - 2 + 52 \times \log P$
- **Speedup** = $(6 \times N - 2)/(6 \times N/P - 2 + 52 \times \log P)$
- **Efficiency** = $(6 \times N - 2)/((6 \times N/P - 2 + 52 \times \log P) \times P) = (6 \times N - 2)/(6 \times N - 2 \times P + 52 \times P \times \log P)$

c) Calculate the speedup and efficiency assuming that the problem for number of processors $p=1$ is that of computing the dot product for two vectors of length 256. Use $p=1, 4, 16, 64, 256$, and assume the same time costs as in (b). [10pts]

**Answer:**

- **Sequential runtime** = $6 \times 256 - 2 = 1534$
- **Parallel runtime:**
  - $P=4$ : $4 \times 256/4 + 2 \times 256/4 - 2 + 52 \times 2 = 486$
  - $P=16$ : $4 \times 256/16 + 2 \times 256/16 - 2 + 52 \times 4 = 302$
  - $P=64$ : $4 \times 256/64 + 2 \times 256/64 - 2 + 52 \times 6 = 334$
  - $P=256$ : $4 \times 256/256 + 2 \times 256/256 - 2 + 52 \times 8 = 420$

  **Efficiency:**
  - $P=4$ : $1534 / 486 / 4 = .789$
  - $P=16$ : $1534 / 302 / 16 = .317$
  - $P=64$ : $1534 / 334 / 64 = .071$
  - $P=256$ : $1534 / 420 / 256 = .014$

d) When executing on 64 processors, how large would $n$ have to be to achieve the same efficiency as achieved on 4 processors for $n=256$? [5pts]
Answer:

\[ \frac{.789}{.09375N + 310} = \frac{6N - 2}{6N/64 - 2 + 52 * 6}/64 \]
\[ \frac{.789}{.09375N + 310} = \frac{6N - 2}{6N/64/63 + 52 * 6}/64 \]
\[ .789 * 64 * (6N/64 - 2) / (.093 * N + 310) = 6N - 2 \]
\[ 50.496 * (.093 * N + 310) = 6N - 2 \]
\[ 4.696128N + 15653.760 = 6N - 2 \]
\[ 4.696128N + 15655.760 = 6N \]
\[ 15655.760 - 4.696128N = 6N - 2 \]
\[ 15655.760/1.303872 = N \]
\[ 12007N. \]

There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There is another theory which states that this has already happened. - Douglas Adams