

1 Thread-Level Parallelism

As powerful as data level parallelization is, it can be quite inflexible, as not all applications have data that can be vectorized. Multithreading, or running a single piece of software on multiple hardware threads, is much more powerful and versatile.

OpenMP provides an easy interface for using multithreading within C programs. Some examples of OpenMP directives:

- The **parallel** directive indicates that each thread should run a copy of the code within the block. If a for loop is put within the block, **every** thread will run every iteration of the for loop.

```
#pragma omp parallel
{
    ...
}
```

NOTE: The opening curly brace needs to be on a newline or **else** there will be a compile-time error!

- The **parallel for** directive will split up iterations of a for loop over various threads. Every thread will run **different** iterations of the for loop. The following two code snippets are equivalent.

<pre>#pragma omp parallel for for (int i = 0; i < n; i++) { ... }</pre>	<pre>#pragma omp parallel { #pragma omp for for (int i = 0; i < n; i++) { ... } }</pre>
--	--

There are two functions you can call that may be useful to you:

- **int** `omp_get_thread_num()` will return the number of the thread executing the code
- **int** `omp_get_num_threads()` will return the number of total hardware threads executing the code

- 1.1 For each question below, state and justify whether the program is **sometimes incorrect**, **always incorrect**, **slower than serial**, **faster than serial**, or **none of the above**. Assume the default number of threads is greater than 1. Assume no thread will complete before another thread starts executing. Assume `arr` is an `int[]` of length `n`.

(a)

```
// Set element i of arr to i
#pragma omp parallel
{
    for (int i = 0; i < n; i++)
        arr[i] = i;
}
```

(b)

```
// Set arr to be an array of Fibonacci numbers.
arr[0] = 0;
arr[1] = 1;
#pragma omp parallel for
for (int i = 2; i < n; i++)
    arr[i] = arr[i-1] + arr[i - 2];
```

(c)

```
// Set all elements in arr to 0;
int i;
#pragma omp parallel for
for (i = 0; i < n; i++)
    arr[i] = 0;
```

- 1.2 What potential issue can arise from this code?

```
1 // Decrements element i of arr. n is a multiple of omp_get_num_threads()
2 #pragma omp parallel
3 {
4     int threadCount = omp_get_num_threads();
5     int myThread = omp_get_thread_num();
6     for (int i = 0; i < n; i++) {
7         if (i % threadCount == myThread) arr[i] -= 1;
8     }
9 }
```

2 Amdahl's Law

In the programs we write, there are sections of code that are naturally able to be sped up. However, there are likely sections that just can't be optimized any further to maintain correctness. In the end, the overall program speedup is the number that matters, and we can determine this using Amdahl's Law:

$$\text{True Speedup} = \frac{1}{S + \frac{1-S}{P}}$$

where S is the non-spiced-up part and P is the speedup factor (determined by the number of cores, threads, etc.).

2.1 You are going to run a convolutional network to classify a set of 100,000 images using a computer with 32 threads. You notice that 99% of the execution of your project code can be parallelized on these threads. What is the speedup?

2.2 You run a profiling program on a different program to find out what percent of this program each function takes. You get the following results:

Function	% Time
f	30%
g	10%
h	60%

- (a) We don't know if these functions can actually be parallelized. However, assuming all of them can be, which one would benefit the most from parallelism?
- (b) Let's assume that we verified that your chosen function can actually be parallelized. What speedup would you get if you parallelized just this function with 8 threads?

3 Warehouse-Scale Computing

Sources speculate Google has over 1 million servers. Assume each of the 1 million servers draw an average of 200W, the PUE is 1.5, and that Google pays an average of 6 cents per kilowatt-hour for datacenter electricity.

3.1 Estimate Google's annual power bill for its datacenters.

3.2 Google reduced the PUE of a 50,000-machine datacenter from 1.5 to 1.25 without decreasing the power supplied to the servers. What's the cost savings per year?

4 Hamming ECC

Recall the basic structure of a Hamming code. We start out with some bitstring, and then add parity bits at the indices that are powers of two (1, 2, 8, etc.). We don't assign values to these parity bits yet. **Note that the indexing convention used for Hamming ECC is different from what you are familiar with.** In particular, the 1 index represents the MSB, and we index from left-to-right. The i th parity bit $P\{i\}$ covers the bits in the new bitstring where the *index* of the bit under the aforementioned convention, j , has a 1 at the same position as i when represented as binary. For instance, 4 is $0b100$ in binary. The integers j that have a 1 in the same position when represented in binary are 4, 5, 6, 7, 12, 13, etc. Therefore, $P4$ covers the bits at indices 4, 5, 6, 7, 12, 13, etc. A visual representation of this is:

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	X		X		X		X		X		X		X		X		X		X		
	p2		X	X			X	X			X	X			X	X			X	X		...
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

Source: https://en.wikipedia.org/wiki/Hamming_code

4.1 How many bits do we need to add to 0011_2 to allow single error correction?

4.2 Which locations in 0011_2 would parity bits be included?

4.3 Which bits does each parity bit cover in 0011_2 ?

4.4 Write the completed coded representation for 0011_2 to enable single error correction. Assume that we set the parity bits so that the bits they cover have even parity.

- 4.5 How can we enable an additional double error detection on top of this?
- 4.6 Find the original bits given the following SEC Hamming Code: 0110111_2 . Again, assume that the parity bits are set so that the bits they cover have even parity.
- 4.7 Find the original bits given the following SEC Hamming Code: 1001000_2

5 RAID

- 5.1 Fill out the following table:

	Configuration	Pro/Good for	Con/Bad for
RAID 0			
RAID 1			
RAID 2			
RAID 3			
RAID 4			
RAID 5			