[20250128] INFOB3CC - Concurrency - 2 - USP

Course: BETA-INFOB3CC Concurrency (INFOB3CC)

Duration:	2 hours
Number of questions:	7
Generated on:	Jun 25, 2025

Contents: Pages:

[20250128] INFOB3CC - Concurrency - 2 - USP

Course: Concurrency (INFOB3CC)

- The exam is a closed book exam.
- The exam must be made alone. No communication with others is allowed.
- Provide brief and concise answers. Overly verbose responses or nonsense added to otherwise good answers can deduct from your grade.
- When asked to explain your choice on a multiple choice question, your reasoning should explain why your chosen answer is correct and why the others are not correct.
- If a question does not give you all the details you need, you may make reasonable assumptions. Your assumptions must be clearly stated. If your solution only works under certain conditions, state them.
- You may assume that threads do not crash.
- You have two hours to complete the exam. You can go back to previous questions.
- *Good luck! (:*

Number of questions: 7

You can score a total of 39 points for this exam, you need 20.29 points to pass the exam.

1 This questions is about the hardware of a typical GPU with lockstep execution.

1 pt. **a.** What set of threads runs together in lockstep execution?

- **a.** All threads in a warp.
- **b.** All threads running together with Simultaneous Multi-Threading.
- **c.** All threads in a thread block.
- **d.** All threads in a grid.
- **e.** All threads on the GPU.

1 pt. **b.** What happens if threads running in lockstep take different paths in an if-then-else statement?

- **a.** It follows the control flow of the first thread. All threads execute only the branch that the first thread takes.
- **b.** It executes both branches. Threads are marked as inactive in one of the two braches.
- **c.** Threads are executed sequentially by a different part of the hardware, not with lockstep execution.
- **d.** This is not supported by most GPUs.

1 pt. **c.** What property or properties of a memory-bound kernel have a direct impact on the occupancy, or the number of threads that can run in parallel on a Streaming Multiprocessor of the GPU?

- **a.** Whether it mixes integer and floating point computations, or only performs one kind of computation.
- **b.** The required amount of shared memory per thread block.
- **c.** The memory access pattern.
- **d.** The number of registers or variables of the program.
- **e.** The required size of global memory.

1 pt. **d.** The GPU has several optimizations to improve the performance of memory. Which of those does one (partially) miss out on, if one performs random reads instead of structured reads?

- **a.** Memory coalescing
- **b.** Warp convergence
- **c.** Caches
- **d.** Determinism
- **e.** Atomicity

1 pt. **e.** Assume we can solve a problem either with random reads and structured writes, or structured reads and random writes. The random reads are not overlapping; we don't read an element twice. The random writes are also not overlapping; we don't write to the same index multiple times.

Which option is likely to be faster?

a. Structured reads and random writes, since the GPU uses cache lines to read values, but not to write values.

b. Structured reads and random writes, since writes can be performed asynchronously.

c. Random reads and structured writes, since random writes would require additional synchronisations.

d. Random reads and structured writes, since a GPU has a large memory bandwidth only for reading values.

e. These perform similarly, since the cost for random reads is similar to the cost of random writes.

2 This question is about parallel scan algorithms on GPUs and the synchronisation methods of these algorithms.

1.5 pt. **a.** A common advice is to ensure that thread blocks are independent. What does this advice mean?

1.5 pt. **b.** How does this make GPU algorithms scalable (to different sizes of GPUs)?

Parallel scans on GPUs split the input into tiles, which are then handled by different thread blocks.

1.5 pt. **c.** How does the reduce-then-scan algorithm communicate information between the different tiles of the array?

1.5 pt. **d.** How does the chained scan algorithm communicate information between the different tiles of the array?

3 In this question, we consider the implementation of depth of field, an image processing effect that blurs the parts of an image that are out of focus. One difficulty in implementing this is that the size of the blur – the number of pixels of the input image that are blurred into one pixel of the output image – differs per pixel. For the pixels in focus, this size will be one (i.e. only the matching input pixel contributes to the output pixel), and for the pixels out of focus this may be a large number. All the input pixels in the neighbourhood given by the blur size should contribute equally, and the shape of the blur is square.

We assume the image is a grayscale image and that color (brightness) is represented as a single number per pixel.

1 pt. **a.** First, we assume that the blur size is the same for all pixels. The output at index (x, y) is the average of all pixels (x', y') where $|x - x'| \leq 3$ and $|y - y'| \leq 3$; each pixel is thus the average of the pixels in the input on a 5 by 5 square.

What is the name of the data-parallel pattern that should be used here?

2 pt. **b.** We implement the blur computation in a single kernel. What is an important optimization for an implementation of the used parallel pattern in this scenario? Give the name of this optimization and explain why it works.

You may ignore the pixels near the edge of the image.

2 pt. **c.** It is possible to make this operation faster by splitting it into two kernels, where the second kernel continues with the output of the first kernel. Explain how and why this will likely make it faster.

You may ignore the pixels near the edge of the image.

In reality, the blur size is not the same for all pixels. For simplicity we will from now on consider a 1-dimensional input. You may also assume that the blur size is always an odd number and that the blur of a pixel never goes out of bounds of the array.

We are now given two input arrays: an array of colors (brightnesses) and an array of blur sizes. Each pixel in the output should now be based on as many pixels as the blur size of that pixel.

As an example, consider the following scenario:

Color	10	0	2	4	8	11	2
Blur size	1	3	3	5	1	3	1
Output	10 / 1 = 10	12 / 3 = 4	6 / 3 = 2	25 / 5 = 5	8 / 1 = 8	21 / 3 = 7	2 / 1 = 2

2.5 pt. **d.** How can this be implemented with the data-parallel patterns that we discussed? Ensure that the speed of your implementation is independent of the blur sizes. You may get fewer or no points if your solution is too slow.

To get more accurate results, we want to specify the blur sizes "the other way round". Previously, we specified for each output pixel how many input pixels it should combine. Here, we instead specify for each *input* pixel, how many *output* pixels it should contribute to.

In the same example, the output would now be:

Color	10	0	2	4	8	11	2
Blur size	1	3	3	5	1	3	1
Output	10 / 1 + 0 / 3	0 / 3 + 2 / 3 + 4 / 5	0 / 3 + 2 / 3 + 4 / 5	2 / 3 + 4 / 5	4 / 5 + 8 / 1 + 11 / 3	4 / 5 + 11 / 3	11 / 3 + 2 / 1

3.5 pt. **e.** How can this form of blurring be implemented with the data-parallel patterns we discussed? Ensure that the speed of your implementation is independent of the blur sizes. You may get fewer or no points if your solution is too slow.

4 Consider the following algorithm, which performs two recursive calls of arrays of half the size and a loop of $\sqrt{n}$ iterations:

```
procedure eagle(ps)
  n = length(ps)

  if n == 0 { return 0 }

  a = eagle(slice(ps, 0, n * 0.5))
  b = eagle(slice(ps, n * 0.5, n))

  c = 0
  for (i = 0; i * i < n; i++) {
    c += ps[i * i]
    ps[i * i] += b
  }

  return c
```

The two recursive calls before the for loop are executed in parallel. Note that the loop terminates when $i * i \geq n$ and thus has $\sqrt{n}$ iterations.

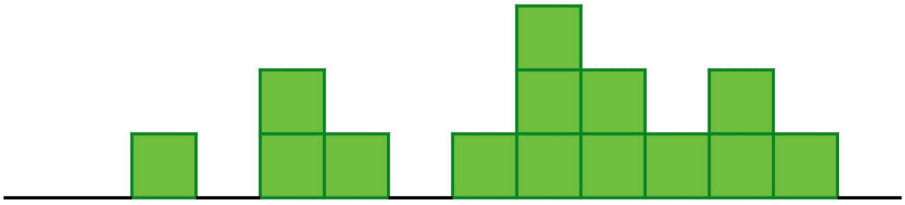
The array length and slice (view the subarray between two indices, without copying) functions can be executed in constant time.

2 pt. **a.** What is the asymptotic span of this algorithm? Show how you computed the span.

2 pt. **b.** What is the asymptotic work of this algorithm? Show how you computed the work.

5
5 pt.

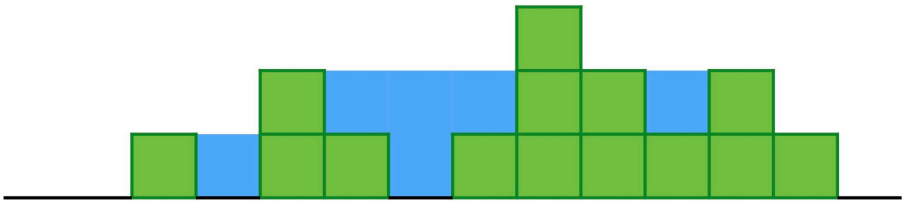
Hilda wants to determine how much water will collect in the terrain after a heavy rainstorm (because her dog loves to swim). She begins with a one-dimensional model of the problem, where she has a vector (one-dimensional array) of integers representing the elevation of that point. The points are recorded at evenly spaced intervals. For example, the following terrain map:



would be given as the vector:

```
elevation = [0, 0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1, 0]
```

This terrain can trap one unit of water in the first basin (between the first block and next, which is 2 blocks high), 4 units of water in the next basin (between the first stack of two blocks and the stack that is three blocks high) and finally one unit of water in the third basin (between the second and third two-block high stacks), as shown below:



Thus the total amount of water that can be held by this terrain is 6 units, which is the result that the function should return.

As the terrain map is very large, we want to use data parallelism to compute the result.

Using the patterns discussed in the lectures (map, stencil, etc.), give a parallel algorithm that will compute the total number of units of water that will be collected in a given terrain.

6 Consider the following attempt to implement a parallel scan over an array *xs* of *n* values:

```
parallel_for (i in 0 .. n) {
  output[i] = parallel_fold(xs, 0, i)
}
```

Function *parallel_fold(xs, a, b)* folds the elements of *xs* between indices *a* (inclusive) and *b* (exclusive) in parallel, with linear work and logarithmic span. The different iterations of the loop are executed in parallel. Note that we assume that the system supports nested data parallelism.

1 pt. **a.** What is the work of this algorithm?

- **a.** $\Theta(1)$
- **b.** $\Theta(\log n)$
- **c.** $\Theta(n / \log n)$
- **d.** $\Theta(n)$
- **e.** $\Theta(n \log n)$
- **f.** $\Theta(n^2 / \log n)$
- **g.** $\Theta(n^2)$
- **h.** $\Theta(n^2 \log n)$

1 pt. **b.** What is the span of this algorithm?

- **a.** $\Theta(1)$
- **b.** $\Theta(\log n)$
- **c.** $\Theta(n / \log n)$
- **d.** $\Theta(n)$
- **e.** $\Theta(n \log n)$
- **f.** $\Theta(n^2 / \log n)$
- **g.** $\Theta(n^2)$
- **h.** $\Theta(n^2 \log n)$

1 pt. **c.** What is the overhead of this algorithm?

a. $\Theta(1)$

b. $\Theta(\log n)$

c. $\Theta(n / \log n)$

d. $\Theta(n)$

e. $\Theta(n \log n)$

f. $\Theta(n^2 / \log n)$

g. $\Theta(n^2)$

h. $\Theta(n^2 \log n)$

0.5 pt. **d.** Is this algorithm efficient?

a. No

b. Yes

0.5 pt. **e.** Is this algorithm optimal?

a. Yes

b. No

7 Charles has developed a sequential algorithm to predict the weather, which runs in $\Theta(n^2)$ time. Tina has the idea for a new algorithm which can be parallelised to run in $\Theta(n \log(n))$ steps, at the expense of increasing the work to $\Theta(n^3)$.

2 pt. **a.** Charles thinks that their company will not be able to afford the processors necessary to make the new algorithm worthwhile. How many processors will be required for Tina's algorithm to be worthwhile (asymptotically) over Charles' algorithm? Motivate your response.

2 pt. **b.** What is the maximum number of processors that the company should purchase for Tina's algorithm (asymptotically)? Motivate your response.

Name:			Signature:
Date:	/ /	Date of birth:	/ /

Course: BETA-INFOB3CC Concurrency (INFOB3CC) - Questions: [20250128] INFOB3CC - Concurrency - 2 - USP

- The exam is a closed book exam.
- The exam must be made alone. No communication with others is allowed.
- Provide brief and concise answers. Overly verbose responses or nonsense added to otherwise good answers can deduct from your grade.
- When asked to explain your choice on a multiple choice question, your reasoning should explain why your chosen answer is correct and why the others are not correct.
- If a question does not give you all the details you need, you may make reasonable assumptions. Your assumptions must be clearly stated. If your solution only works under certain conditions, state them.
- You may assume that threads do not crash.
- You have two hours to complete the exam. You can go back to previous questions.
- *Good luck! (:*

1
5 pt.

a. A ○ B ○ C ○ D ○ E ○

b. A ○ B ○ C ○ D ○

c. A ☐ B ☐ C ☐ D ☐ E ☐

There are multiple answers possible

d. A ☐ B ☐ C ☐ D ☐ E ☐

There are multiple answers possible

e. A ○ B ○ C ○ D ○ E ○

2
6 pt.

a.

Answer:

b.

Answer:

c.

Answer:

d. Answer:

2

4

6

8

3
11 pt.

a.

Answer:

2

b.

Answer:

2

4

6

8

10

12

c. Answer:

d. Answer:

e. Answer:

4
4 pt.

a.

Answer:

b.

Answer:

5
5 pt.

Answer:

2

4

6

8

6
4 pt.

a. A ○ B ○ C ○ D ○ E ○ F ○ G ○ H ○

b. A ○ B ○ C ○ D ○ E ○ F ○ G ○ H ○

c. A ○ B ○ C ○ D ○ E ○ F ○ G ○ H ○

d. A ○ B ○

e. A ○ B ○

7
4 pt.

a.

Answer:

b.

Answer:

Elaboration of the answer

1. 5 pt.

a. 1 pt. A

b. 1 pt. B

c. B
D

d. A
C

e. 1 pt. C

2. 6 pt.

a.

Correction criterion	Points
Within a kernel,	0 to 0.5 points
the different threadblocks don't communicate / don't synchronize / can be executed independently from each other	0 to 1 points
Total points:	*1.5 points*

b.

Correction criterion	Points
The hardware is free to assign blocks to any processor (SM) at any time	0 to 0.8 points
This can scale to any number of SMs (if there are sufficient thread blocks)	0 to 0.7 points
Total points:	*1.5 points*

c.

Correction criterion	Points
The algorithm is split into multiple kernels	0 to 0.5 points
The scan phase (last kernel) only starts when all work of the reduce phase (first kernel) is finished.	0 to 1 points
Total points:	*1.5 points*

d.

Correction criterion	Points
It blocks within a kernel	0 to 0.5 points
After the reduce-phase, a thread block waits (blocks) on the result of the previous thread block	0 to 1 points
Total points:	*1.5 points*

3.
11 pt.

a.

Correction criterion	Points
stencil	0 to 1 points
Total points:	*1 point*

b.

Correction criterion	Points
Tiling / strip mining	0 to 1 points
Compute output in an order to reuse values that we already read (in cache)	0 to 1 points
Total points:	*2 points*

c.

Correction criterion	Points
stencil 5x1 after stencil 1x5 (or the other way around)	0 to 1 points
Fewer memory operations and/or calculations	0 to 1 points
Total points:	*2 points*

d.

Correction criterion	Points
Integral image / integralImg = scanl (+) 0 colors	0 to 1 points
backpermute / gather (or imap / generate) to read integralImg !! (idx + blurSize / 2) and integralImg !! (idx - blurSize / 2)	0 to 1 points
Subtract these and divide by blurSize	0 to 0.5 points
Total points:	*2.5 points*

e.

Correction criterion	Points
zipWith (/) color blurSize	0 to 0.7 points
permute or scatter	0 to 0.8 points
permute to add each color to the location where it starts contributing (idx - blurSize/2)	0 to 0.5 points
permute to subtract each color from the location where it stops contributing (idx + blurSize/2 + 1)	0 to 0.5 points
scanl (+) 0	0 to 1 points
Total points:	*3.5 points*

4.
4 pt.

a.

Correction criterion	Points
f(n) = sqrt(n) = n^0.5	0 to 0.5 points
T(n) = T(n/2) + sqrt(n)	0 to 0.5 points
Case 3 OR: f dominates over recursive calls	0 to 0.5 points
Span is T(n) = O(sqrt(n))	0 to 0.5 points
Total points:	*2 points*

b.

Correction criterion	Points
f(n) = sqrt(n)	0 to 0.5 points
T(n) = 2T(n/2) + sqrt(n)	0 to 0.5 points
Case 1 OR: recursive calls dominate over f	0 to 0.5 points
Span is T(n) = O(n)	0 to 0.5 points
Total points:	*2 points*

5.
5 pt.

Correction criterion	Points
An element of the array can store water if there are higher bars on the left and the right	1 point
Compute the maximum height seen so far on the left: maxl = scanl1 max elevation	1 point
Compute the maximum height seen so far on the right: maxr = scanr1 max elevation	1 point
Given these pre-computed arrays, the amount of water stored at each point: zipWith3 (\l r h -> min l r - h) maxl maxr elevation	1 point
Sum the result of the previous step: fold (+) 0	1 point
Total points:	*5 points*

6.
4 pt.

a. 1 pt. G

b. 1 pt. B

c. 1 pt. D

d. 0.5 pt. A

e. 0.5 pt. B

7.
4 pt.

a.

Correction criterion	Points
The break even point is at P = Wt/Sc = n^3/n^2 = n. With less than this number of processors, the sequential algorithm is faster.	0 to 2 points
Total points:	*2 points*

b.

Correction criterion	Points
The transition between the work bound and span bound phase is P = work / span = n^3 / n log(n) = n^2 / log n.	0 to 2 points
Total points:	*2 points*

Caesura

Applied guessing score: 1.59 pt

Points scored	Grade
39	10
38	9.76
37	9.52
36	9.28
35	9.04
34	8.80
33	8.56
32	8.32
31	8.08
30	7.83
29	7.59
28	7.35
27	7.11
26	6.87
25	6.63
24	6.39
23	6.15
22	5.91
21	5.67
20	5.43
19	5.19
18	4.95
17	4.71
16	4.47
15	4.23
14	3.99
13	3.75
12	3.50
11	3.26
10	3.02
9	2.78
8	2.54
7	2.30
6	2.06
5	1.82
4	1.58
3	1.34
2	1.10
1	1.00
0	1.00

Question identifiers

These identifiers can be used to track the exact origin of the question. Use these identifiers together with the identifier of this document when sending in comments about the questions, so that your comment can be connected precisely with the question you are referring to.

Document identifier: 67540-116292

Question number	Question identifier	Version identifier
1	651499	31205e70-c0c9-44b0-9b36-730e796b5f53
2	651573	64215e3e-f09b-4332-a8c6-ea7a4185bf9f
3	651301	8e9fc5d3-7f54-4991-b73f-52fddf935f20
4	494833	15013f8c-16c9-42cb-892b-ad98b1e36040
5	497918	588c4a2b-ee3a-6fc0-08c2-ef3189be61d3
6	651488	6df7fce3-963d-4af6-bd6b-c5334b481e81
7	520509	929e4075-f93e-4326-a0f9-e0b783d4ce8f