Solutions

Marking: **M1** a correct method, **A1** the correct answer, **B1** an independent mark.

Each answer shows the steps, then the **result**.

2.1

$$\frac{1}{0.2 + 0.8/4}$$

$= \frac{1}{0.4} =$ **2.5**

limit $1/0.2 =$ **5**

2.2

$512/16 =$ **32** values

one instruction does 32 additions

2.3

Gustafson: $(1 - p) + pN$

$0.05 + 0.95 \times 64$

$=$ **60.85**

2.4

both numbers are in one cache block

each write makes the other core's copy invalid, so the block moves back and forth (false sharing); fix by padding them into separate blocks

3.1

(a) $10 + 90/16 = 15.625$ s; $100/15.625 =$ **6.4**

(b) time $100/8 = 12.5$ s, so $10 + 90/N = 12.5$; $N =$ **36**

(c) $100/10 =$ **10**

(d) $6.4/16 =$ **40**%

3.2

(a) `count += 1` is a read, an add and a write; if both threads read the same value before either writes, one update is lost

(b) hold a lock around the update, or use an atomic add

(c) each thread counts in its own local variable and adds it to `count` once at the end, so the lock is taken twice instead of 2×10^6 times

3.3

(a) $8 \times 8 \times 3 \times 10^9 = 1.92 \times 10^{11}$ operations a second; $10^{10}/1.92 \times 10^{11} \approx$ **0.052** s

(b) compute $10^{10}/10^{13} = 0.001$ s; data 4×10^8 B, 0.025 s each way; total $\approx$ **0.051** s

exercises.lol exercises.com.cn

(c) CPU $10 \times 0.052 =$ **0.52** s; GPU $10 \times 0.001 + 0.05 =$ **0.06** s

(d) one filter: either, because the transfers cancel the GPU's gain, so the CPU is simpler; ten filters: the GPU, about 9 times faster, because the transfer cost is paid once