1 Question 1: Amdahl's Law

1.1 Task a)

Disregarding the misplaced marketing hyperbole for the sake of the argument and assuming the new train route follows the bus route (i.e. has the same length), we designate the section from Luzern to Rotkreuz as serial fraction: $f_1 = \frac{55\text{km}-36\text{km}}{55\text{km}} = 0.345$. This leaves the potential new train section as improvable fraction: $f_p = 1 - f_1 = 0.655$. Based on the information given, we assume that the average speed $v_b = \frac{55\text{km}}{1.5\text{h}} = 36.67\text{km/h}$ of the bus on the total leg LU-ZH also applies to the fraction f_1, whereas $v_t = 70\text{km/h}$ is given for the train speed on the fraction f_p. This in turn gives a speedup factor $p = \frac{70\text{km/h}}{36.67\text{km/h}} = 1.91$.

1.1.1 Task a.i)

Nobody will get any improvement from the announcement per se, but if implemented, it will offer a speedup of 1.453 as seen from equation 1. In other words, the ideal one-way journey time will reduce from 90 minutes to 62 minutes.

$$S_p = \frac{1}{f_1 + \frac{f_p}{p}} = \frac{1}{0.345 + \frac{0.655}{1.91}} = 1.453 \tag{1}$$

1.1.2 Task a.ii)

Again using equation 1, this time with a (rounded) speedup factor of $p = 3 \cdot 10^7$, we get a theoretical speedup of 2.9 and thus a one-way journey time of about 30 minutes.

1.2 Task b)

As per the task description, we disregard hyperthreading and thus plainly assume 12 CPU cores per socket. Also, we are given that $f_p = 0.9$ and thus $f_1 = 1 - f_p = 0.1$. Finally, we assume that the code currently is in serialized form running on a single core.

1.2.1 Task b.i)

The desired speedup is given as $S_p = 8$. Solving equation 1 for p, we get that the desired speedup from serial execution requires a minimum of 36 cores (equation 2).

$$p = \frac{S_p \cdot f_p}{1 - S_p \cdot f_1} = \frac{8 \cdot 0.9}{1 - 8 \cdot 0.1} = 36 \tag{2}$$

1.2.2 Task b.ii)

As we need three full sockets (36 / 12 = 3), but sockets only come paired up as nodes, we require two nodes. As we then might as well use them completely, the obtainable speed up is, again using equation 1, $S_{p=48} = \frac{1}{0.1+\frac{0.9}{48}} = 8.42$.

1.2.3 Task b.iii)

The price for additional nodes can safely be assumed to be non-negligible. It can therefore credibly be argued that buying more nodes doesn't make any sense, as obviously even for $p \to \infty$ the absolute best obtainable speedup would be $\lim_{p\to\infty} S_p = \frac{1}{0.1+\frac{0.9}{p}} = 10$.

2 Question 2: Manual Vectorization of Reduction Operator

2.1 Task a)

Done as instructed.

2.2 Task b)

Done as instructed.

2.3 Task c)

Done as instructed.

2.3.1 Task c.i)

In accordance with the SIMD width, a speed up of 4 and 2 respectively was expected for serial execution. With thread level parallelism, the maximum additional speedup factor should be in the order of the number of threads employed.

2.3.2 Task c.ii)

The differences between lasrge and small vector size seem somewhat inconclusive. However it is evident that the overhead in the OpenMP implementation as expected didn't result perfect speedup.

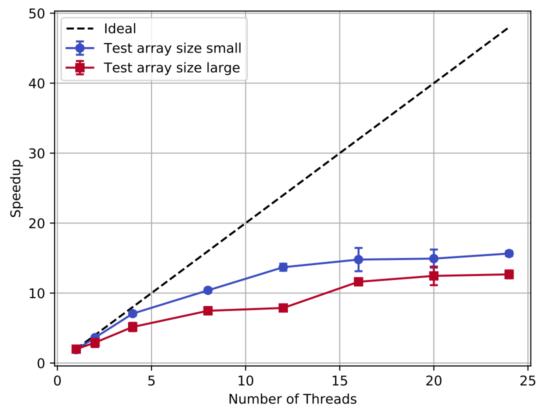

Figure 1: Thread level parallelism speedup of reduction operator: Double precision data.

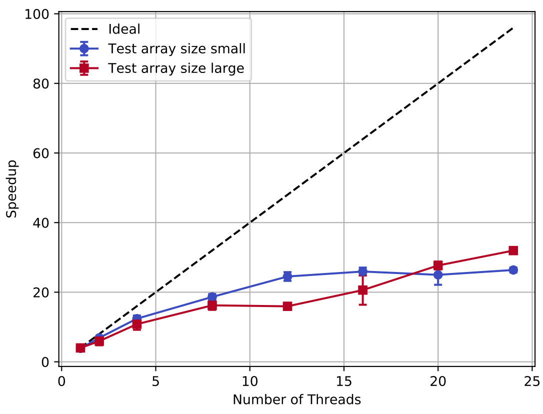

Figure 2: Thread level parallelism speedup of reduction operator: Single precision data.

3 Question 3

3.1 Task a)

Done as instructed.

3.2 Task b)

Done as instructed.

3.3 Task c)

Attempted as instructed, however ran out of time when trying to properly implement `ISPC` matrix multiplication.

3.4 Task d)

See c) above; implementation not fully functional.

3.5 Task e)

See c) above; implementation not fully functional.

3.6 Task f)

See c) above; implementation not fully functional.