1.2.1

split the program into two part: parallel fraction p
serial fraction s

So $s + p = 1$

normalize the execution time in 1 unit $T_1 = 1$

if we have N parallel execution units

total time $T_N = s + \frac{p}{N}$

speedup $a(N) = \frac{T_1}{T_N} = \frac{1}{s + \frac{p}{N}}$

as we know $s = 1 - p \Rightarrow a(N) = \frac{1}{(1-p) + \frac{p}{N}}$

1. 2. 2

1 The corretness of Amdahl’s Law in read applications

Amdahl’s Law is a theoretical upper bound. But in real applications, the program usually does not meet its assumptions, so the law sometimes predicts accurately, sometimes it is too high, and sometimes it is too low. ③

2 Predict too high

When the real systems have additional overhead

Such as communication costs, memory contention

3 Predict too low

When the problem size is not fixed, but increase. Then we can find more part to do in parallel ways.

Such as big data machine learning with more gpus, MapReduce

1.2.3

(a) $N = 4, \quad p = 0.5$

$$a = \frac{1}{(1-0.5) + \frac{0.5}{4}} = 1.6$$

(b). $N = 8, \quad p = 0.2$

$$a = \frac{1}{(1-0.2) + \frac{0.2}{8}} = \frac{1}{0.825} \doteq 1.21$$

(c). it told us, if p is very small, increase N can not speed up much time.

increase p is more effective than add N.