Multicore Computing Homework 2

Wenwen Zhang
Chunheng Luo

September 3, 2016

Question 0

TACC User IDs:

- Chunheng Luo: chluo
- Wenwen Zhang:

Question 1

Part (a)

- Assuming other parts of the program can be sped up by the factor of n, then the overall speedup is

$$Speedup = \frac{1}{0.4 + \frac{0.6}{n}}$$

- Assuming the method M accounts for x of the program's execution time on a single-core processor, M can be sped up by 2^3 and other parts of the program can be sped up by the factor of n, then the overall speedup is

$$Speedup = \frac{1}{\frac{x}{8} + \frac{(1-x)}{n}}$$

So in order to double the speedup, we require

$$\frac{1}{\frac{x}{8} + \frac{(1-x)}{n}} = 2 \times \frac{1}{0.4 + \frac{0.6}{n}}$$

which leads to

$$x = \frac{0.2n - 0.7}{0.125n - 1}$$

Therefore, M must account for $\frac{0.2n-0.7}{0.125n-1}$ of the total execution time **on a single-core processor** in order to double the overall speedup of the program.

Part (b)

Assuming the parts of the program that can be totally parallelized account for P of the total execution time on a single-core processor, and all of the other parts of the program, which accounts for $(1-P)$ of the total execution time on a single-core processor, are not able to gain any speedup from the multicore architecture, then we have

$$S_2 = \frac{1}{(1-P) + \frac{P}{2}}$$

and

$$S_n = \frac{1}{(1-P) + \frac{P}{n}}$$

Solving the equations, we get

$$S_n = \frac{nS_2}{(2-n)S_2 + 2(n-1)}$$

Question 2

Question 3

In order make Filter Algorithm able to solve the l-exclusion problem, we can simply reduce the number gates from N to $(N - l)$.

```
const int  N;                  // N processors
int[N]     gate init 0;        // Need N - l gates
int[N-l+1] last init 0;        // The proc that gets stuck at each gate

/* For P_i */
request CS;
for (k = 1 : N - l) {
        gate[i] = k;            // P_i is at gate k now
        last[k] = i;            // P_i updates last for that gate

        int forward = l + 1;    // Number of threads ahead of P_i
        while ( (forward >= l) && (last[k] == i) ) {
            forward = 0;
                for (j = 1 : N - 1) {
                if ( (j != i) && (gate[j] >= k) )
                        forward++;
                }
                NO_OP();
        }
}
CS;
release CS;
gate[i] = 0;
```

Question 4

Question 5