Homework 1

PANGBO

1.10

(a)

Original workload is $100,000$, for a 100 times faster system, the workload can be done in the same time is $100,000 \times 100 = 10,000,000$.

So the new problem size is $10,000,000$.

(b)

Original workload is $100,000 \log_2 100,000$, for a 100 times faster system, the workload can be done in the same time is $100,000 \log_2 100,000 \times 100 = 10,000,000 \log_2 100,000$.

Denote the new problem size as n, then we have $n \log_2 n = 10,000,000 \log_2 100,000$. Solve this equation, we have $n = 7,285,986$.

(c)

Original workload is $100,000^2$, for a 100 times faster system, the workload can be done in the same time is $100,000^2 \times 100 = 10^{12}$.

The new problem size is $\sqrt{10^{12}} = 1,000,000$.

(d)

Original workload is $100,000^3$, for a 100 times faster system, the workload can be done in the same time is $100,000^3 \times 100 = 10^{17}$.

The new problem size is $\sqrt[3]{10^{17}} = 464,158$.

1.13

Data parallelism:

(1) In the second row, 3 As can be computed in parallel.
(2) In the fourth row, 3 As can be computed in parallel.

Functional parallelism:

(1) In the third row, the computation of B and C can be done in parallel.
(2) In the fourth row, the computation of D and 3 As can be done in parallel.

Author's address: pangbo.

2.18

(a)

The directory should be distributed among the multiprocessor's local memories for scalability. As the number of processors increases, the centralized directory will become a bottleneck. Moreover, some directory entries may be accessed more frequently by some processors, distributed directory can help to improve performance.

(b)

The contents of the directory are not replicated to prevent data inconsistency and overhead. If the directory is replicated, the directory entries may be inconsistent, and the overhead of maintaining the consistency is high.

17.2

(a)

```
#pragma omp parallel for
for (i = 0; i < (int) sqrt(x); i++) {
    a[i] = 2.3 * i;
    if (i < 10) b[i] = a[i];
}
```

(b)

This code segment is not suitable for parallel execution because the loop termination condition depends on the flag variable

(c)

```
#pragma omp parallel for
for (i = 0; i < n; i++) {
    a[i] = foo(i);
}
```

(d)

```
#pragma omp parallel for
for (i = 0; i < n; i++) {
    a[i] = foo(i);
    if (a[i] < b[i]) a[i] = b[i];
}
```

(e)

This code segment is not suitable for parallel execution because there is a break statement in the loop, so the total number of iterations is not known in advance.

(f)

```
dotp = 0;
#pragma omp parallel for reduction(+:dotp)
for ( i = 0; i < n; i++)
    dotp += a[i] * b[i];
```

(g)

```
#pragma omp parallel for
for ( i = k; i < 2*k; i++)
    a[i] = a[i] + a[i-k];
```

(h)

This code segment is not suitable for parallel execution because the calculation of a[i] depends on the value of a[i-k], but there is no guarantee that a[i-k] has been calculated before a[i].

17.4

Increasing the chunk size in an interleaved scheduling of tasks can improve the cache hit rate due to the principle of spatial locality. When pragmas wants to write data to memory that is not in the cache, CPU will allocate a cache line to store the data. If the data access pattern is spatially local, the data that will be accessed in the near future is likely to be in the same cache line, so the cache hit rate will be improved.

If the chunk size is set to 1, a thread will write to a different memory location in each iteration, which brings two main problems:

(1) First, each thread will write to a different memory location in each iteration, which will cause a high cache miss rate and a high memory writeback frequency.
(2) Second, it is more likely that same memory block to be write by different threads simultaneously, which will cause cache coherence problem and brings large overheads to maintain the cache consistency.

17.6

```
#pragma omp parallel for private(j) schedule(dynamic)
for (i = 0; i < n; i++) {
    int k = rand();
    for (j = 0; j < k; j++) {
        a[i] = a[i] + j;
    }
    }
```

Since the time to execute the inner loop is not known in advance, the dynamic scheduling is more suitable for this code segment.