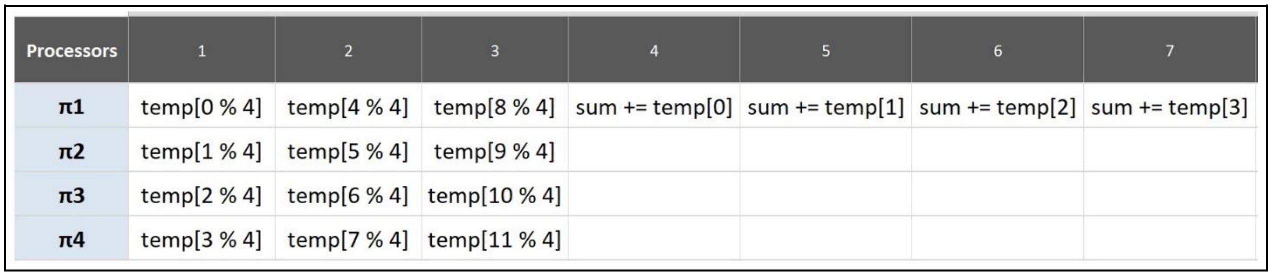
Uzochi Dimkpa
Code Redesign
ITCS 5145–001
Parallel Computing
Prof. Erik Saule
February 14, 2023

1.1

```
int reduce (int* array, size_t n, size_t p) {
        int temp_arr [p]; int sum = 0;
        for (int i = 0; i < n; i++) {
                temp_arr[i % p] = array[i];
        }
        for (int i = 0; i < p; i++) {
                sum += temp_arr[i];
        }
        return sum
}
```

- $\boldsymbol{width} = \boxed{p}, \sum p_i = \boxed{n+p}, T_\infty = \boxed{p + \frac{n}{p}}$
- **4 processors:** $n = 12, p = 4,$



Processors	1	2	3	4	5	6	7
π1	temp[0 % 4]	temp[4 % 4]	temp[8 % 4]	sum += temp[0]	sum += temp[1]	sum += temp[2]	sum += temp[3]
π2	temp[1 % 4]	temp[5 % 4]	temp[9 % 4]				
π3	temp[2 % 4]	temp[6 % 4]	temp[10 % 4]				
π4	temp[3 % 4]	temp[7 % 4]	temp[11 % 4]				

1.2

- No, because for a maximum value to be chosen, a comparison of values has to be made, which would increase the work, width, and critical path.
- Yes, because concatenating strings is just adding them onto one another.
- Yes, because even if the type of values being added changed, the operation is still the same, so the function remains virtually identical.
- No, because for a maximum value to be chosen, a comparison of values has to be made, which would increase the work, width, and critical path. Even if the value types being compared were floats, the fact that they need to be compared adds an entirely new dimension to the function that is not accounted for in this version of the parallel program.

Uzochi Dimkpa
Code Redesign
ITCS 5145–001
Parallel Computing
Prof. Erik Saule
February 14, 2023

2

```
void prefixsum (int* arr, int n, int* pr) {
        return prefixsum(arr[:n/2], n/2, pr[n/2]) + prefixsum(arr[n/2:], (n - n/2), pr[n\2:])
}
```

- $\boldsymbol{width} = \boxed{\boldsymbol{n}}, \sum \boldsymbol{p_i} = \boxed{\boldsymbol{n + log(n)}}, \boldsymbol{T_\infty} = \boxed{\boldsymbol{log(n)}}$

3

- …
- $\boldsymbol{width} = \boxed{\cdots}, \sum \boldsymbol{p_i} = \boxed{\cdots}, \boldsymbol{T_\infty} = \boxed{\cdots}$
- **Parallelized merge sort:** $\boldsymbol{width} = \boxed{\cdots}, \sum \boldsymbol{p_i} = \boxed{\cdots}, \boldsymbol{T_\infty} = \boxed{\cdots}$