

Assignment 2: Extracting Parallelism

Due Sep 8

The purpose of this assignment is for you

- to develop insight about how to extract parallelism from simple codes.
- to recognize cases where some form of parallelism may not be correct
- to acknowledge that sometimes adding work is necessary

Note: Often when talking about algorithm, the weights of tasks are denoted with their complexity.

Note: 1 and 2 are exercise/warm up. 3, 4 and 5 are harder.

1 Transform

Consider the transform function:

```
void transform (int* a, int* b, int n) {  
    for (int i=0; i<n; ++i)  
        b[i] = f(a[i]);  
}
```

Question: Extract the dependencies. Assume the call to `f` cost $O(1)$.

Question: What is the width? the critical path? the work?

Question: How does a schedule look like on P processors.

2 Reduce

Consider the reduce function:

```
template<typename T, typename op>  
T reduce (T* array, size_t n) {  
    T result = array[0];  
    for (int i=1; i<n; ++i)  
        result = op (result, array[i]);  
    return result;  
}
```

So if you define `T` as `int` and `op` as `sum`, it boils down to computing the sum of the array. You could use `op` as `max` and compute the maximum value of the array.

2.1 int, sum

Consider first the `int, sum` case which computes the sum of an array of integer.

Question: Extract the dependencies of this problem. What is the width? the critical path? the work?

Question: Noticing that the different loop iteration could execute in any order. Introduce a mutual exclusion cause on the dependency graph. Does that help?

Question: Rewrite the code to introduce one local variable per processor to store partial computation. Extract the dependencies now. What is the width, critical path and work ?

Question: What does a schedule look like on P processors?

2.2 Variants

Question: Would these two parallel versions (with mutual exclusion and with local variable) be correct for `int, max`?

Question: Would these two parallel versions (with mutual exclusion and with local variable) be correct for `string, concat`?

Question: Would these two parallel versions (with mutual exclusion and with local variable) be correct for `float, sum`?

Question: Would these two parallel versions (with mutual exclusion and with local variable) be correct for `float, max`?

3 Find first

3.1 in an array

Question: Write a sequential algorithm that search a value `val` in an array `arr` of size `n` and return the position `pos` of the first location where `arr[pos] == val`. (and returns `n` otherwise.)

Question: What is the complexity of this algorithm? (as a function of `pos` and `n`).

Question: Can you make a parallel algorithm with $\theta(n)$ work? What is its critical path and width?

Question: Can you make a parallel algorithm with $\theta(pos)$ work? What is its critical path and width?

3.2 in a linked list

Question: What do you think about doing the same thing with a linked list?

4 Prefix sum

Prefixsum is the algorithm that computes $pr[i] = \sum_{j \leq i} arr[j]$ and often written sequentially:

```
void prefixsum (int* arr, int n, int* pr) {  
    pr[0] = arr[0];  
    for (int i=1; i<n; ++i)  
        pr[i] = pr[i-1] + arr[i];  
}
```

Question: What is the structure of the dependency of prefixsum?

Question: How can you make it parallel? (Hint: you have to add work, a single path is not enough)

5 Merge Sort

Question: Recall the merge sort algorithm

Question: Extract dependency on the merge sort algorithm. What is the critical path, work, and width?

Question: How does the schedule of such an algorithm look like when $P=4$?

Question: Can you extract more parallelism? (Hint: increase the amount of work slightly)