Parallel Programs

What is parallelization and why?

What is parallelization and why?

- Conceptual reason
  - Sometimes it give you conceptual isolation among parallel units
- **Performance** reason
  - True parallelism: get multiple CPUs running at the same time
  - Concurrency: keep the CPU utilization high, while some concurrent units are waiting for I/Os
- Modern hardware/system trend
  - Multicore computers
  - Distributed systems

How to parallelize a sequential algorithm?

Data parallelization

Task parallelization

Pipeline parallelization

What code can (not) be executed in parallel?

Principle

- Parallel running code should have little dependence with each other

- When there is dependence
  - Synchronization is needed → slowdowns
  - Without synchronization → concurrency bugs (races)

Examples

- Matrix addition
- Array summation
- Array sorting
    - Quicksort
    - Mergesort
    - Bubblesort

Matrix addition

- How to parallelize it?

Matrix addition

- How to parallelize it?

- Use data paralelizatoin
  - When the same operation is applied on many different variables/data, we can make the operation for different data execute in parallel
  - Suppose we have K CPU cores, we can make each core work on N/K rows (N is the dimension of the matrix row)
  - What if we make each core work on N/K columns?

Array Summation

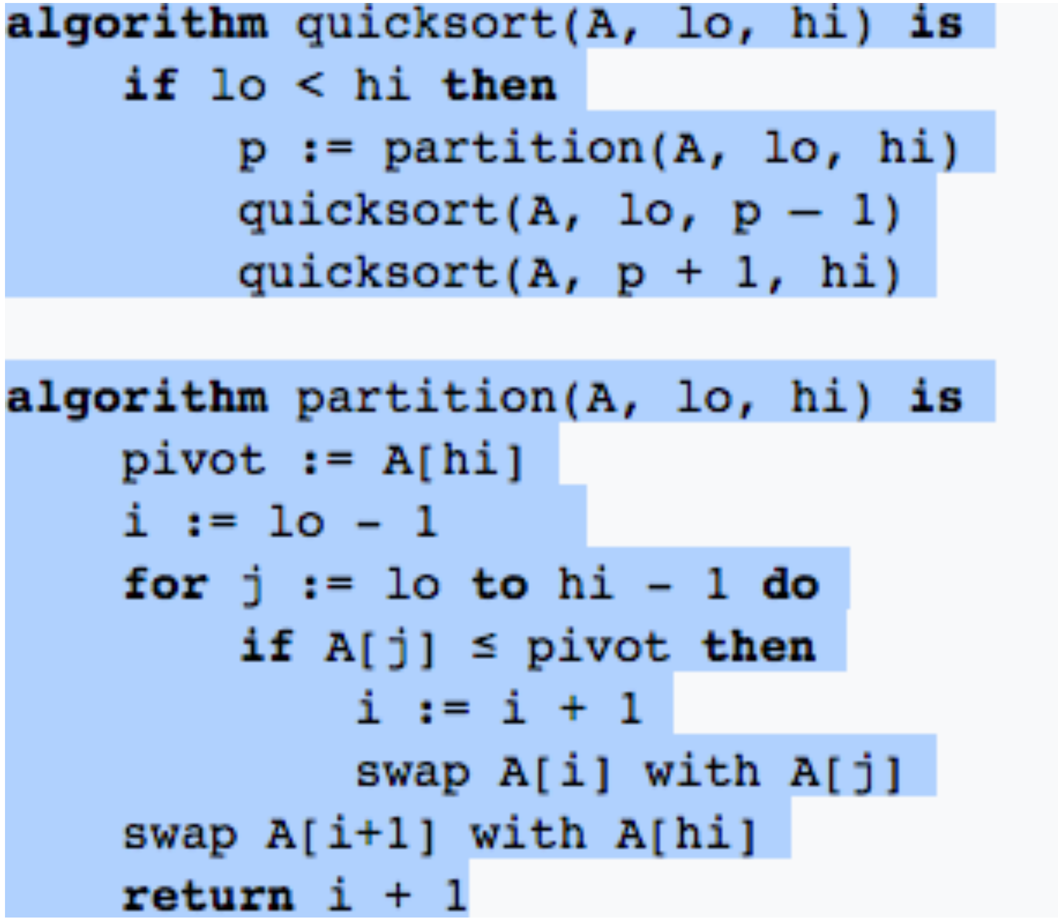
```
int sum = 0;
for (int i=0; i< M; i++)
    sum = sum + A[i];
printf (“sum is %d”, sum);
```

Array summation

- Sometimes, we need to change the sequential code a little bit …

```
for (I =0 ; i<M/4; i++)
   sum1 = sum1 + A[i]
For (i=M/4; I <M/2; i++)
   sum2 = sum2 + A[i]
…
…
Sum1+sum2+sum3+sum4
```

Quicksort



```
algorithm quicksort(A, lo, hi) is
    if lo < hi then
        p := partition(A, lo, hi)
        quicksort(A, lo, p – 1)
        quicksort(A, p + 1, hi)

algorithm partition(A, lo, hi) is
    pivot := A[hi]
    i := lo - 1
    for j := lo to hi - 1 do
        if A[j] ≤ pivot then
            i := i + 1
            swap A[i] with A[j]
    swap A[i+1] with A[hi]
    return i + 1
```

How to parallelize quicksort?

- Run the two quicksort in parallel

- What if we have more than 2 CPUs?
- Are we guaranteed to get 2X speedup?

Mergesort

Divide the unsorted list into n sublists, each containing 1 element

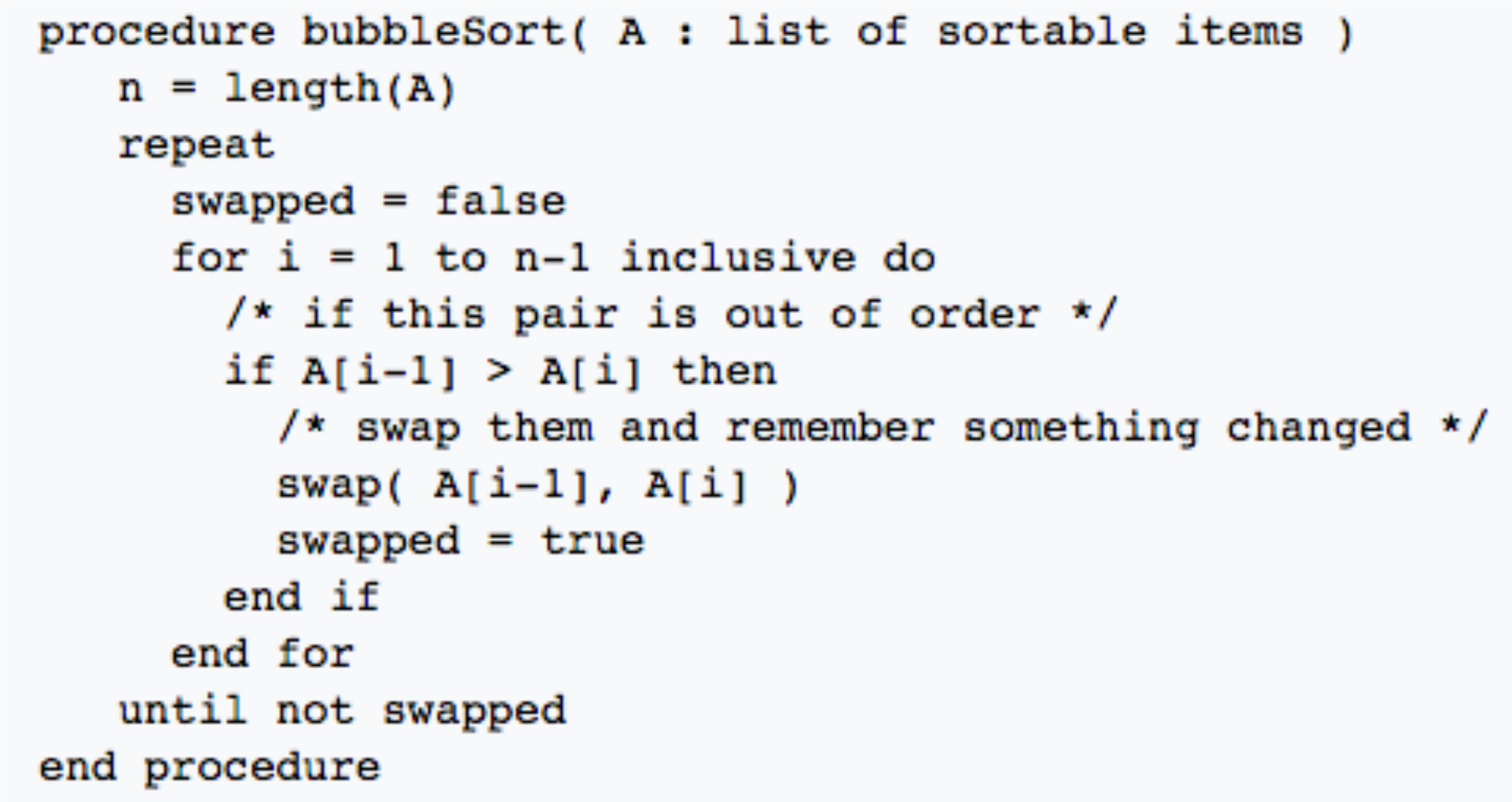
Repeatedly merge sublists to produce new sorted sublists, until there is only 1 sublist remaining

How to parallelize merge-sort?

- Run the merge sort on different sub-lists in parallel

- Merge-sort is among the easiest to parallize sorting algorithms

Bubble sort



```
procedure bubbleSort( A : list of sortable items )
   n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i-1] > A[i] then
         /* swap them and remember something changed */
         swap( A[i-1], A[i] )
         swapped = true
       end if
     end for
   until not swapped
end procedure
```

Bubble sort

- Bubble sort is extremely difficult to parallelize because there are strong dependency among loop iterations

Example summary

- Matrix addition
  - Trivial data parallelism
  - Pay attention to row/column memory layout
- Array summation
  - Easy data parallelism, but we cannot follow the original sequential implementation where there is dependency among loop iterations
  - Cut the array to sub-arrays, get sub-array sum, aggregate
- Array sorting
  - Quicksort
  - Mergesort
  - Bubblesort

A more difficult example

```
while (! End of source file)
    read a line
    process the line
    write the processing result to destination file
```

How to parallelize?

Use pipeline parallelism: run three threads as following

CPU1: Read line 1 → Process line 1 → write result 1 → read line 4 → process line 4

CPU2: read line 2 → process line 2 → write result 2 → read line 5 → …

CPU3: read line 3 → process line 3 → write result 3 → read line 6 →

How to parallelize a sequential algorithm?

- Data parallelization
- Task parallelization
- Pipeline parallelization