Fork - Join Pattern

A Short Introduction

Joseph Kehoe[1]

[1]Department of Computing and Networking
Institute of Technology Carlow

CDD101, 2017

Table of Contents

Table of Contents

Definition

- Pattern used to implement Map, Reduce, Recurrance and Scan (amongst others)
- Uses a Divide and Conquer Approach
- Recursively break a problem up into subproblems until base case is reached
- Base Case is amount of work too small to be parallelised further (problem dependent)
- It is important that the subproblems are independent

Pseudo-code

```
void divide&Conquer(P)
{
        if (P is BASE CASE){
          solve P;
        } else {
          Divide P into K subproblems
          foreach  P[i]:0<=i<K{
            fork(divide&Conquer(P[i]))
          }
          join
          Combine all partial solutions P[i]
        }
 }
```

Speed

- if there are K divisions of the problem at each recursive level
- and there are N levels of recursion
- then we get K^N-way parallelism
- getting the no. of levels N correct is important (and problem dependent)

Table of Contents

Simple OpenMP Implementation

```
#pragma omp task
fnA();
fnB();
#pragma omp taskwait
```

Notes

- taskwait acts as join point or "barrier"
- task causes next line of code to run in parallel
- We can use {} to make block of code run in parallel
- Must have pragma parallel in code before task otherwise it will not run in parallel (explain!)
- Global variables captured by reference (shared)
- Local variables captured by value (firstprivate)

Table of Contents

QuickSort

- Pick an element, called a pivot, from the array
- Partitioning: reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the partition operation.
- Recursively apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

QuickSort Pseudocode

```
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]
    if A[hi] < A[i + 1] then
      swap A[i + 1] with A[hi]
  return i + 1
```

Examples

MergeSort

- Divide the unsorted list into n sublists, each containing 1 element (a list of 1 element is considered sorted)
- Repeatedly merge sublists to produce new sorted sublists until there is only 1 sublist remaining
- This will be the sorted list.