CS3383 Unit 4: dynamic multithreaded algorithms

David Bremner

March 15, 2024

Dynamic Multithreaded Algorithms

- Fork-Join Model
- Span, Work, And Parallelism
- Parallel Loops

[redacted]

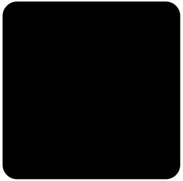
Dynamic Multithreaded Algorithms

Keywords

parallel	for loop iterations are (potentially) concurrent
spawn	Run the procedure (potentially) concurrently
sync	Wait for all spawned children to complete.

See

- CLRS4 Chapter 26
- Cilk, Cilk+
- OpenMP



Fibonacci Example

function $\text{Fib}(n)$
    if $n \leq 1$ **then**
        return n
    else
        $x =$ spawn $\text{Fib}(n-1)$
        $y = \text{Fib}(n-2)$
        sync
        return $x + y$
    end if
end function

Serialization

- removing keywords yields correct serial code
- Adding parallel keywords to correct serial code might break it (e.g. race conditions).

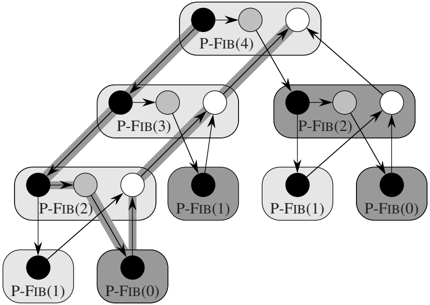
Fibonacci example in OpenMP

```
long fib(int n) {
  long x, y;
  if (n<=1)
    return n;
  else {
    #pragma omp task shared(x)
    x=fib(n-1);
    y=fib(n-2);
    #pragma omp taskwait
    return x+y;
  }
}
```

Computation DAG

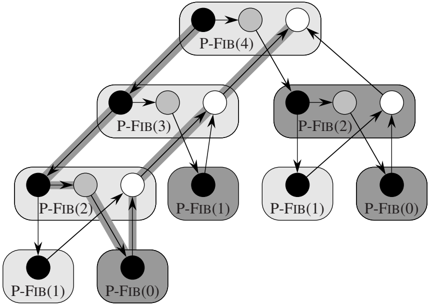
Strands: Sequential instructions with no *parallel*, *spawn*, return from *spawn*, or *sync*.

function $\textsc{Fib}(n)$
  if $n \le 1$ **then** ▷ ●
    return n
  else
    $x =$ spawn $\text{Fib}(n-1)$
    $y = \text{Fib}(n-2)$ ▷ ●
    sync
    return $x + y$ ▷ ○
  end if
end function



Computation DAG

Strands: Sequential instructions with no *parallel*, *spawn*, return from *spawn*, or *sync*.



nodes strands

down edges spawn

up edges return

horizontal edges sequential

span length of longest path

Work and Speedup

- T_1 *Work*, sequential time.
- T_p Time on p processors.

Work Law

$$T_p \geq T_1/p$$

$$\text{speedup} := T_1/T_p \leq p$$

Parallelism

T_p Time on p processors.

T_∞ *Span*, time given unlimited processors.

We could idle processors:

(1) $$T_p \geq T_\infty$$

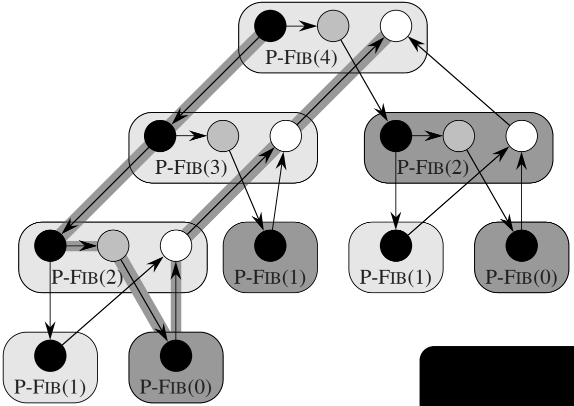
Best possible speedup:

$$\text{parallelism} = T_1/T_\infty \geq T_1/T_p = \text{speedup}$$

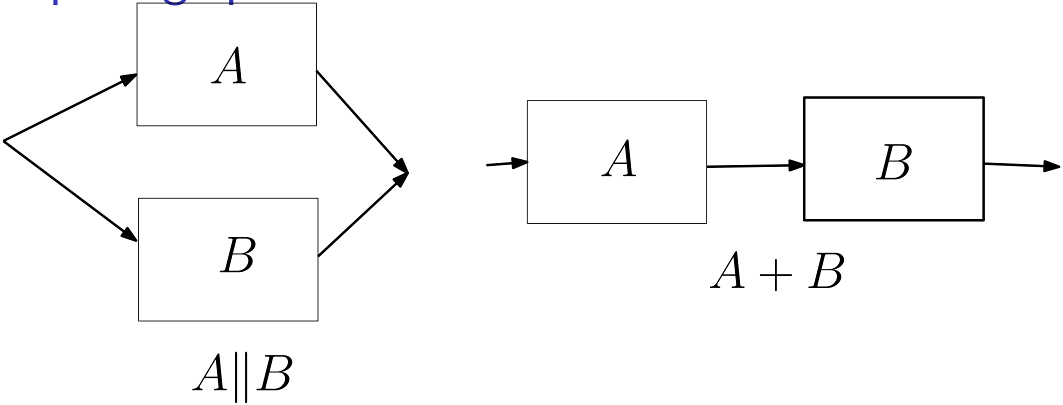
Span and Parallelism Example

Assume strands are unit cost.

- $T_1 = 17$
- $T_\infty = 8$
- Parallelism $= 2.125$ for this input size.



Composing span and work



series $T_\infty(A+B) = T_\infty(A) + T_\infty(B)$

parallel $T_\infty(A\|B) = \max(T_\infty(A), T_\infty(B))$

series or parallel $T_1 = T_1(A) + T_1(B)$

Work of Parallel Fibonacci

Write $T(n)$ for T_1 on input n.

$$T(n) = T(n-1) + T(n-2) + \Theta(1)$$

Let $\phi \approx 1.62$ be the solution to

$$\phi^2 = \phi + 1$$

We can show by induction (twice) that

$$T(n) \in \Theta(\phi^n)$$

Span and Parallelism of Fibonacci

$$
\begin{aligned}
T_\infty(n) &= \max(T_\infty(n-1), T_\infty(n-2)) + \Theta(1) \\
&= T_\infty(n-1) + \Theta(1)
\end{aligned}
$$

Transforming to sum, we get

$$T_\infty \in \Theta(n)$$

$$\text{parallelism} = \frac{T_1(n)}{T_\infty(n)} = \Theta\left(\frac{\phi^n}{n}\right)$$

- inefficient, but very parallel

Parallel Loops

parallel for $i = 1$ to n **do**

    statement...

    statement...

end for

- Run n copies in parallel with local setting of i.
- Effectively n-way spawn
- Can be implemented with spawn and sync
- Span

$$T_\infty(n) = \Theta(\log n) + \max_i T_\infty(\text{iteration i})$$

Parallel Matrix-Vector product

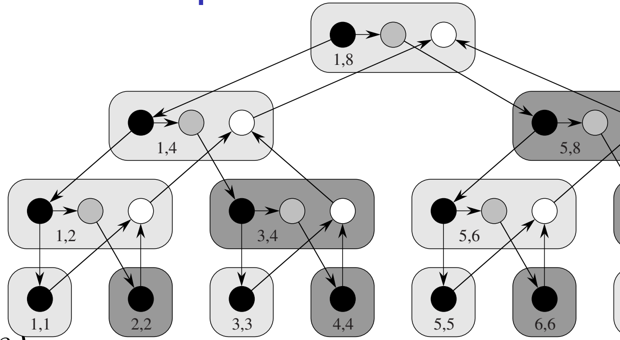
function ROWMULT(A,x,y,i)
    $y_i = 0$
    for $j = 1$ to n **do**
        $y_i = y_i + a_{ij}x_j$
    end for
end function

- Why is RowMult not using parallel for?

function MAT-VEC(A, x, y)
    Let $n =$ rows(A)
    parallel for $i = 1$ to n **do**
        RowMult(A,x,y,i)
    end for
end function

Divide and Conquer Matrix-Vector product

function $\mathrm{MVDC}(A, x, y, f, t)$
    if $f == t$ **then**
        RowMult(A,x,y,f)
    else
        $m = \lfloor (f+t)/2 \rfloor$
        spawn MVDC(A, x, y, f, m)
        MVDC$(A, x, y, m+1, t)$
        sync
    end if
end function



Divide and Conquer Matrix-Vector product

function $\mathrm{MVDC}(A, x, y, f, t)$
    if $f == t$ **then**
        RowMult(A,x,y,f)
    else
        $m = \lfloor (f+t)/2 \rfloor$
        spawn MVDC(A, x, y, f, m)
        MVDC$(A, x, y, m+1, t)$
        sync
    end if
end function

- $T_\infty(n) = \Theta(\log n)$ (binary tree) + leaf cost
- $\Theta(n)$ leaves (one per row)
- $\Theta(n)$ interior nodes (binary tree)
- $T_1(n) = \Theta(n^2)$