Umut Acar [2018/07/09] 10:45AM (1)

(Lecture 1)

umutaca@gmail.com
umut@cmu.edu

Parallelism is "hard" ... or is it?

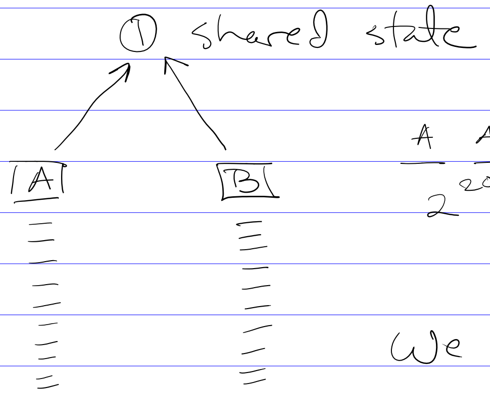
Language-based cost models!

- abstract
- truthful

"CS should be as advanced as restaurant business"

Assume: Pure Functional Programs

no shared (mutable) state



A A B A B B _ _ _

2^{20} possible interleaving of instructions

We can't reason about correctness of such programs.

$\boxed{\lambda\text{-calc}}$ is a good model for writing and reasoning about programs.

Cost semantics

simple, w/ a little twist: we care not only about work, but also "span".

Cost = Work ← sequential runtime (additive)
+ Span ← parallel (max)

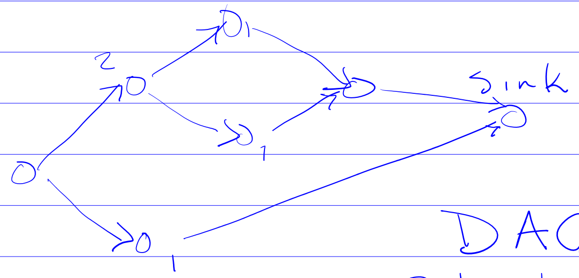
Example

```
fun f x = if x ≤ 1 then x
          else let (a,b) = (f(x-1) || f(x-2))
               in a+b end
```

$$W(n) = \begin{cases} 1 & \text{if } n \le 1 \\ W(n-1) + W(n-2) + 1 & \text{o/w} \end{cases}$$

$$S(n) = \begin{cases} 1 & \text{if } n \le 1 \\ \text{Max}\,(S(n-1), S(n-2)) & \text{o/w} \end{cases}$$

$n = 3$



DAG (dir. acyclic graph)

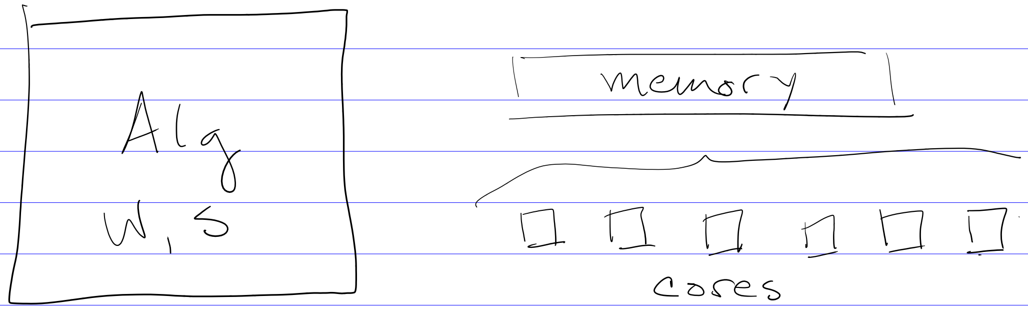
But only a subset of DAGs are relevant (series-parallel DAGs)

Want
- Work Efficient
- Low Span

How do we make our abstract cost model TRUTHFUL?

Provably Efficient Implementations

Bounded Implementation



Parallel algorithm should agree with sequential semantics. (i.e. The result should be the same as when doing it sequentially)

Lower Bounds: $T_p \ge W/p \quad T_p \ge S$

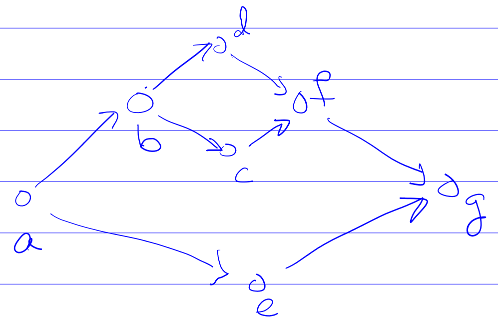
Determining $T_p^{OPTIMAL}$ is NP-complete.

However, approximating the optimal is easy.

Umut Acar (Lect 1) [2018/07/09] (11 30 AM)

Schedule (for example above)

	1	2
1	a	
2	b	c
3	d	e
4	f	
5	g	



(N.B. This is not breadth-first.)

Let w_i = # vertices at level i

$$W = \sum_{i=1}^{S} w_i$$

$$T_p = \sum_{i=1}^{S} \left\lceil \frac{w_i}{p} \right\rceil = \sum \left\lfloor \frac{w_i}{p} \right\rfloor + 1 = \left(\sum \left\lfloor \frac{w_i}{p} \right\rfloor \right) + S$$

$$\leq \sum \frac{w_i}{p} + S = \frac{W}{p} + S$$

(This is Brent's Theorem (1974))

Claim: $T_p \leq \frac{W}{p} + S \leq 2 * T^{OPT}$

$$\left. \begin{array}{l} T_p \geq \frac{W}{p} \\ T_p \geq S \end{array} \right\} \quad T_p \geq \max\left(\frac{W}{p}, S\right)$$

so $T_p^{OPT} \geq \max\left(\frac{W}{p}, S\right)$

(The cost of the scheduler vs Length of the schedule (T_p)) we'll come back to this.

Greedy scheduling

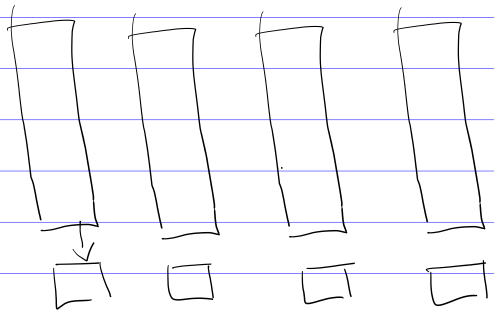
If $\exists$ idle processors $\wedge$ $\exists$ ready vertices then assign.

Theorem: $T_p \leq \frac{W}{p} + S \cdot \frac{(p-1)}{p}$

$$T_p = \frac{\text{total tasks left}}{p} \leq \frac{W + (p-1) \cdot S}{p}$$

Umut Acar (Lect 1) [2018/07/09] 11:53AM

Distributed Queues



Work stealing for Load Balancing

If proc. is idle, Try to steal work from other proc. queues.

Now our algorithm is concurrent instead of parallel. Very hard to implement correctly and efficiently.

Probabilistic Algorithm with expected cost

$$E[T_P] \le \frac{W}{P} + S \quad \text{(includes cost of scheduler)}$$