Comp 212: Functional Programming

Dan Licata

Spring 2021

Parallelism

Hardware

1. Multicore CPUs / processor (+)

OLD: CPUs do one thing at a time

NEW: Many laptop 2 cores
4 cores
8 cores
16 cores

(2) GPUs (graphics)

↳ 1000s of things at once

(3) Data centers

10,000s of computers

16-32-64 cores

Parallel Functional Programming

- Say what is to be computed at a high level
- let the compiler assign work to processors

```
(* Purpose: count the number of
        people who took 211 in
        Python *)
type class = row sequence
fun count(c : class) : int =
    sum (2) (map sum c) (1)
```

int sequence

(1) in parallel, sum each row

(2) sum the column at the end

```
(* Purpose: add up a row *)
type row = int sequence

sum : row -> int

(* Example:
     sum <1,1,0,1,1> == 5 *)
```

Computing by calculation

Val row1 : row = <1, 1, 0, 1, 1>

Val row2 : row = <0, 1, 1, 0, 1>

Val class = <row1, row2>

Count class

runs by stepping through the definitions of the functions

count ⟨row1, row2⟩

$\longmapsto$ sum (map sum ⟨row1, row2⟩)

$\longmapsto$ sum (⟨sum row1, sum row2⟩)

$\longmapsto$ sum (⟨ 4 , 3 ⟩)

$\longmapsto$ 4+3

$\longmapsto$ 7

mathematical definition of what it means to run a program

You will learn to:

1. write parallel functional programs
2. analyze their sequential and parallel running times

Suppose classroom n students in n rows

$n \times n$

Sequential: $\approx n^2$ $O(n^2)$

(work) → 1 processor

parallel: $\approx n$ $n \approx n$

step 1 + step 2

(span) → "enough" processors

infinite

Work and span predict

the running time with

P processors

for any P

via

Brent's Principle

time on p processors

$$\approx \max\left(\frac{\text{Work}}{p}, \text{Span}\right)$$

$n = 10$ 100 students

Work = 100

Span = 10

$p = 2$ $\frac{\text{work}}{p} = 50$

$$\max\left(\frac{W}{p}, S\right) = \max(50, 10)$$

$$= 50$$

$P = 100$ $\quad \frac{W}{P} = 1$

$S = 10$

$Max(1, 10) = 10$

You will learn to:

1. write parallel functional programs
2. analyze their sequential and parallel running times
3. reason mathematically about their correctness
4. structure large programs using abstract types

sum: int sequence -> int

↓

somfaster: int sequence -> int

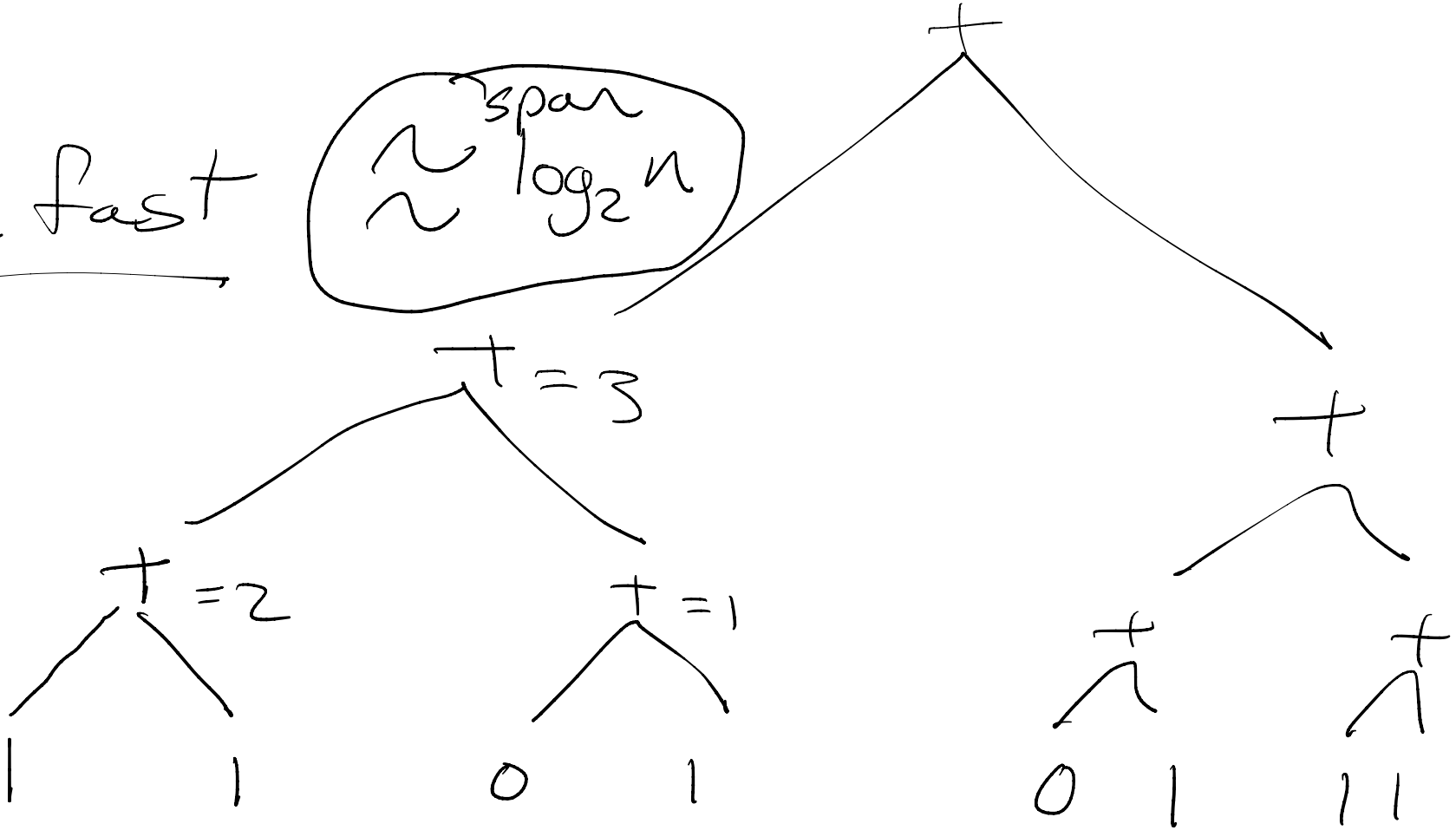
Theorem: for all s: int sequence,

sum(s) == somfast(s)

Sum has span $\approx n$ (length of row)

$1+(1+(0+1))$

Sum fast



Activities

1. pre-recorded / asynch lectures
2. labs weekly
   - Th 1pm
   - Th 2:50 pm
   - Fri 1pm
3. Homework — Wed 1pm
   + help sessions
   - → due Thursday before labs (?)

Assessment

5% labs

95% homework

- regular → help
- challenge → no help