CS61A Lecture 9 MapReduce

2011-07-05

Eric Tzeng

iClicker activity!

Vote for whichever letter you think will have the *least* votes by the time class starts!

Feel free to ask each other what you're voting for (but obviously you don't have to tell the truth!)

Administrivia

- Project 2 due tonight by 11:59pm!
- Review session this Saturday (7/9)
  - 1:00 to 4:00 pm
  - 306 Soda
- Midterm 1 next Monday (7/11)
  - 7:30 to 10:00 pm
  - 2050 VLSB

Reminder: `accumulate`

```
STk> (trace accumulate)
STk> (accumulate + 0 '(1 2 3 4))
(+ 1 (accumulate + 0 '(2 3 4)))
(+ 1 (+ 2 (accumulate + 0 '(3 4))))
(+ 1 (+ 2 (+ 3 (accumulate + 0 '(4)))))
(+ 1 (+ 2 (+ 3 (+ 4 (accumulate + 0 '())))))
(+ 1 (+ 2 (+ 3 (+ 4 0))))
(+ 1 (+ 2 (+ 3 4)))
(+ 1 (+ 2 7))
(+ 1 9)
10
```

Reminder: `accumulate` (con't)

```
STk> (trace accumulate)
STk> (accumulate cons '() '(a b c))
(cons 'a (accumulate cons '() '(b c)))
(cons 'a (cons 'b (accumulate cons '() '(c))))
(cons 'a (cons 'b (cons 'c (accumulate cons '() '()))))
(cons 'a (cons 'b (cons 'c '())))
(cons 'a (cons 'b '(c)))
(cons 'a '(b c))
'(a b c)
```

Reminder: `accumulate` (con't)

- Potential asymmetry in arguments:

```
(accumulate
  (lambda (next so-far) (+ next so-far))
  0
  '(1 2 3))
```

always a number | always a number

```
(accumulate
  (lambda (next so-far) (cons next so-far))
  '()
  '(1 2 3))
```

always a number | always a list ← Arguments are not the same type!

Higher Order Function Review!

For these questions, use only higher order functions!

1. Write a procedure `product-of-squares` that takes a list and returns the product of the squares of each element.

```
STk> (product-of-squares '(1 2 3))
36
```

2. Write a procedure `letter-count` that counts the number of letters in a list of words.

```
STk> (letter-count '(hello there))
10
```

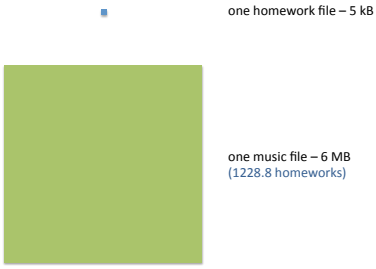
Intro to MapReduce

2005 – 200 terabytes of information indexed

200 Terabytes?

one homework file – 5 kb
(5×10^3 bytes)

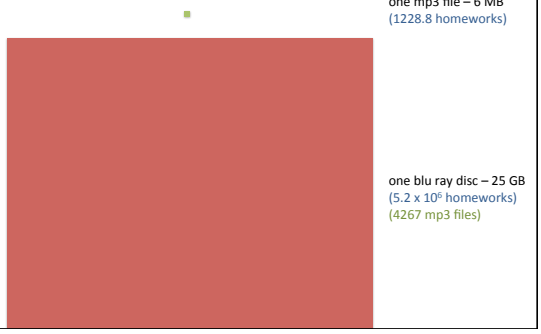
200 Terabytes?



one homework file – 5 kB

one music file – 6 MB
(1228.8 homeworks)

200 Terabytes?

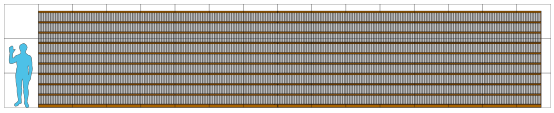


one mp3 file – 6 MB
(1228.8 homeworks)

one blu ray disc – 25 GB
(5.2×10^6 homeworks)
(4267 mp3 files)

Tangent: how big is Wikipedia?

Wikipedia is only 16 gigabytes!



200 Terabytes?

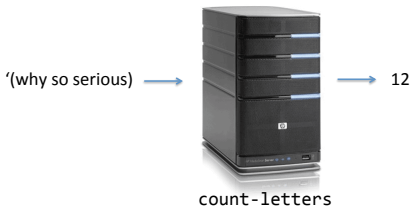
- one blu ray disc – 25 GB
  (5.2 x 10^6 homeworks)
  (4267 mp3 files)

Google's index size as of 2005 – 200 TB
(42 x 10^6 homeworks)
(35 x 10^6 mp3 files)
(8192 blu ray discs)

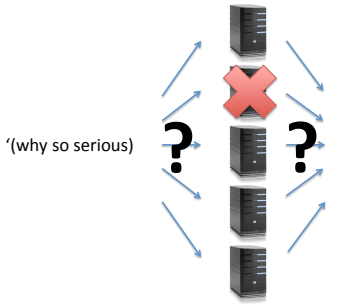
"Between the birth of the world and 2003 there were 5 exabytes of information created […] [now] we create 5 exabytes in two days."

- Eric Schmidt, Former Google CEO, 2008

The normal way



The parallel way



Splitting up and managing the work is hard!

Let's take a look at that again…

```
(define (product-of-squares ls)
  (accumulate * 1
    (map square ls)))

(define (letter-count ls)
  (accumulate + 0
    (map count ls)))
```

The basic idea

```
(accumulate + 0
  (map count '(why so serious)))
```

(kinda) MapReduce

The basic idea (map phase)

```
(accumulate + 0
  (map count '(why so serious)))
```

The "mapper"
- Procedure of one argument

The basic idea (reduce phase)

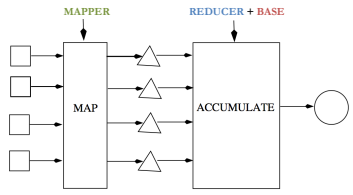
The "reducer"
- Procedure of two arguments

base-case
- The final value passed to the reducer

```
(accumulate + 0
  (map count '(why so serious)))
```

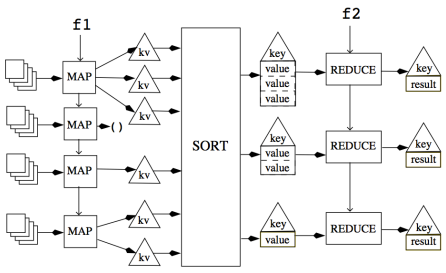
The basic idea

```
(define (kinda-mapreduce mapper reducer base input)
  (accumulate reducer base
    (map mapper input)))
```



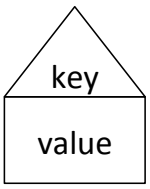
Break time!

mapreduce



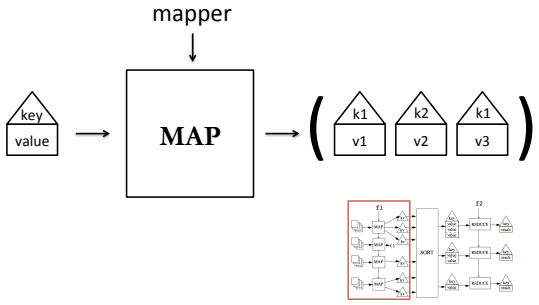
Key-Value Pairs

```
(define make-kv-pair cons)

(define kv-key car)

(define kv-value cdr)
```



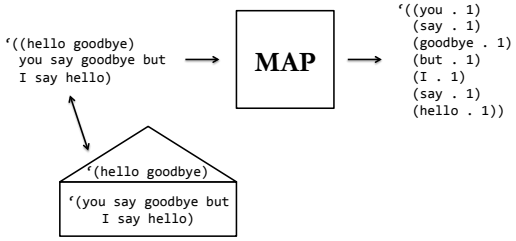
Reminder: respect the abstraction!

mapreduce – map phase

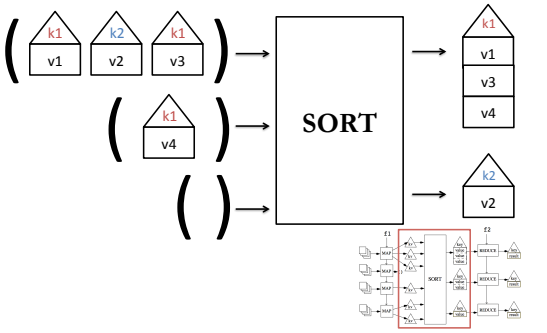


map phase - example

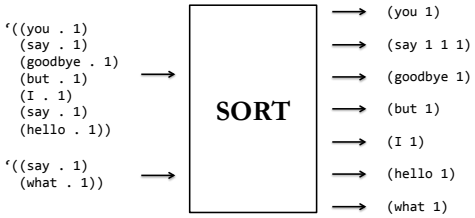
What mapper will perform this transformation?



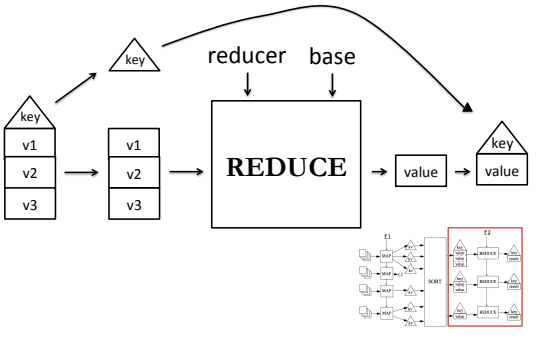
mapreduce – sort phase



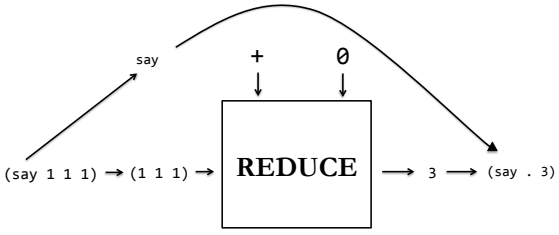
sort phase - example



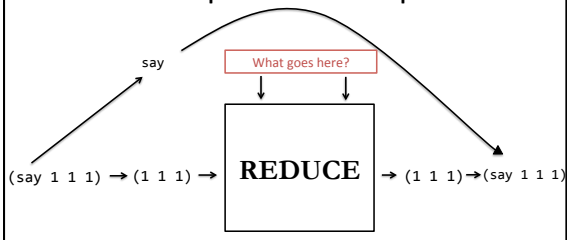
mapreduce – reduce phase



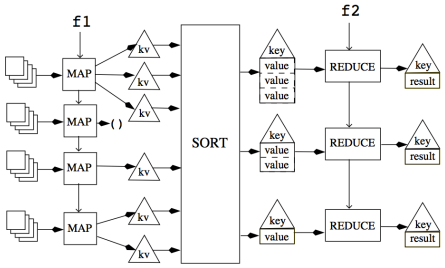
reduce phase - example



reduce phase – example 2



mapreduce



Go over all the parts one more time with your neighbor!

True (A) or false (B)?

1. The reducer is a procedure that takes a list of values and combines them.
2. The mapper can act as a filterer.
3. The kv-pairs that the mapper produces will have the same keys as the final output kv-pairs.

mapreduce example

Input:

a set of kv-pairs of the form

```
((play title) . (line from that play))
```

Output:

the number of letters in each play, not counting the word "forsooth"

"Forsooth, a mapper!"

```
(define (forsooth-mapper kvp)
  (map
    (lambda (wd) (make-kv-pair (kv-key kvp) wd))
    (filter
      (lambda (wd) (equal? wd 'forsooth))
      (kv-value kvp))))
```

"Alack! A reducer!"

```
(define (forsooth-reducer next so-far)
  (+ next so-far))

;; alternatively...
(define forsooth-reducer +)
```

Putting it all together

- All that's left to do is to make our final call:

```
STk> (mapreduce forsooth-mapper            ;; mapper
                forsooth-reducer           ;; reducer
                0                          ;; base-case
                "/gutenberg/shakespeare")  ;; input
```

Why mapreduce?

mapper reducer base-case

above the line

below the line

Splitting up work

Process management

Failure handling

etc…