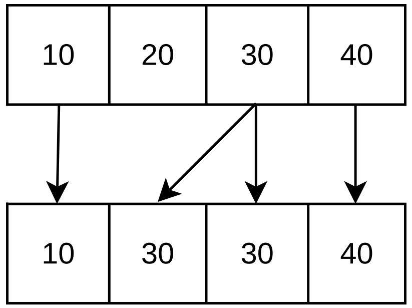
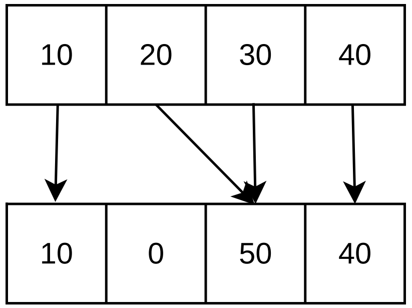
Nested Data Parallelism

Irregular data parallelism

Non-uniform access



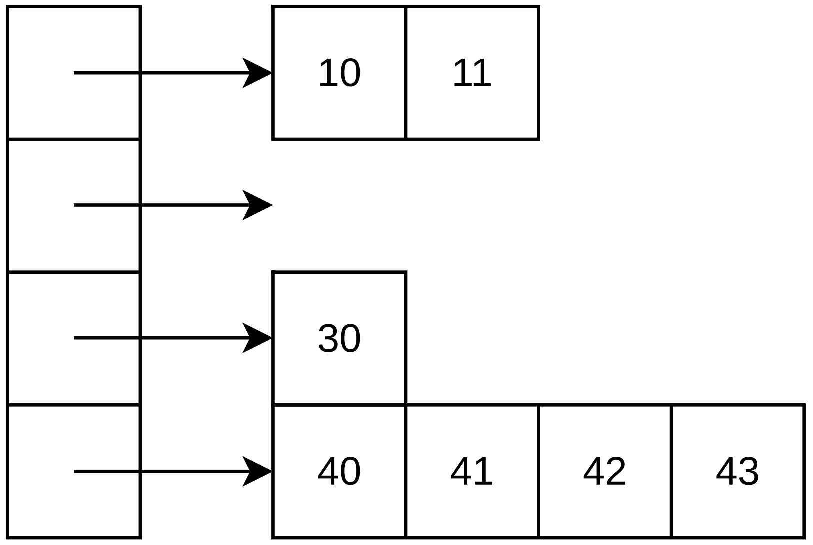
Gather



Scatter

Nested Data Parallelism

Ragged arrays

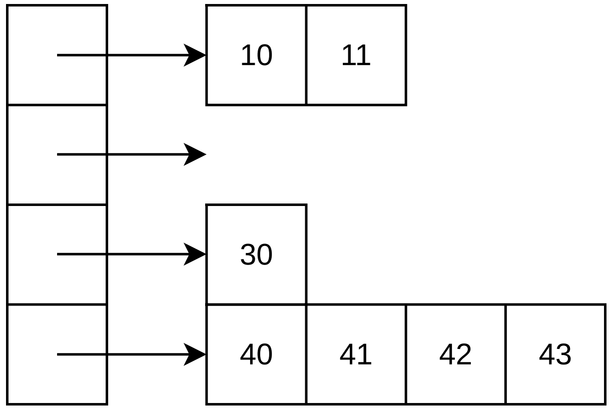


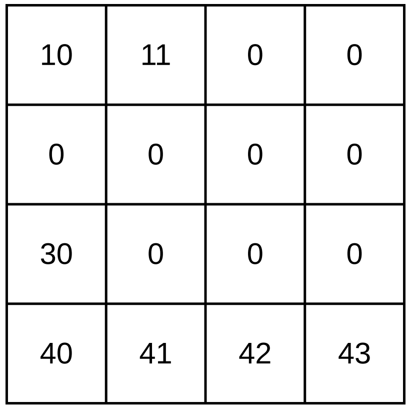
When to use it?

Whenever you have structured data with differently sized components.

Examples: sparse matrices, graph algorithms, hierarchical space divisions, document analysis, path tracing, ...

Only when saved work exceeds overhead.





Futhark

https://futhark-lang.org/

> Futhark is a [...] statically typed, data-parallel, and purely functional array language in the ML family, and comes with a heavily optimising ahead-of-time compiler that presently generates either GPU code via CUDA and OpenCL, or multi-threaded CPU code.

```
def average (xs: []f64) = reduce (+) 0 xs / f64.i64 (length xs)
```

Designed by Troels Henriksen, Cosmin Oancea, Martin Elsman at DIKU, from 2014

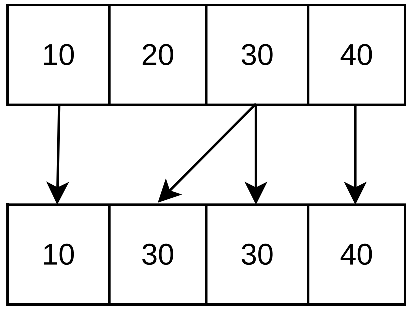
Regular Data Parallelism in Futhark

```
def scale [n] (a: [n]f64): [n]f64 =
  map (*2) a

def add [n] (a: [n]f64) (b: [n]f64): [n]f64 =
  map2 (+) a b

def factorial (n: i64): i64 =
  reduce (*) 1 (1...n)
```

https://futhark-lang.org/docs/prelude/

demo/basics.fut

Gather

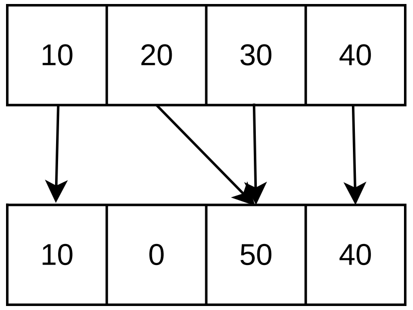


```
def gather (xs: []f64) (is: []i32) =
  map (\i -> xs[i]) is

entry main =
  gather [10,20,30,40] [0,2,2,3]
```

demo/gather.fut

demo/gather2.fut

Scatter

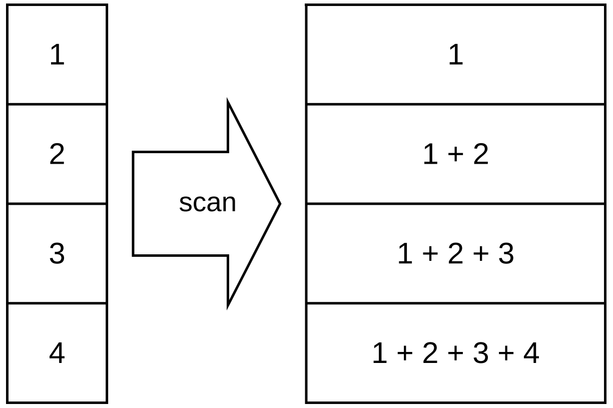


```
def scatter (is: []i64) (vs: []f64) : []f64 =
  reduce_by_index (replicate 4 0) (+) 0 is vs

entry main (is: [4]i64) (vs: [4]f64) : [4]f64 =
  scatter is vs
```

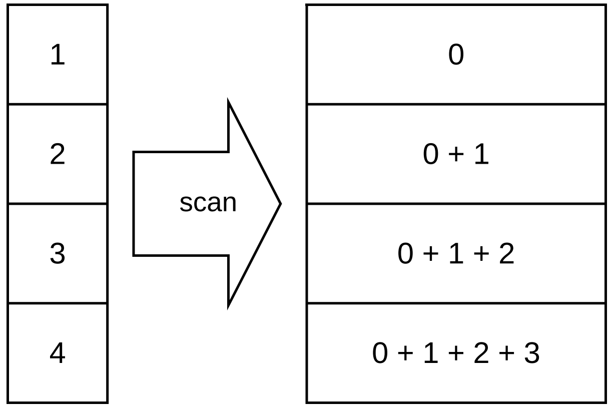
demo/scatter.fut

demo/histogram.fut

Parallel Scan



Inclusive scan



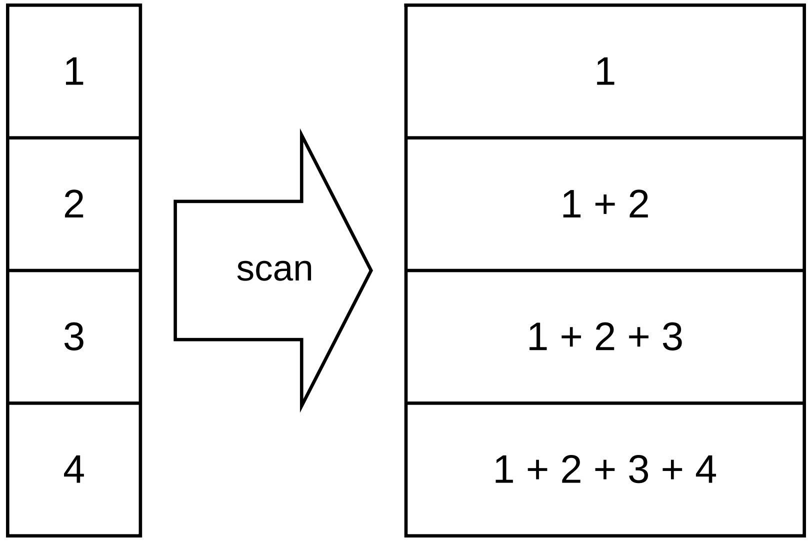
Exclusive scan

When to use it

- integral images
- parsing brackets
- sparse matrices
- radix sort
- ...

Work and Span of Parallel Scan

http://conal.net/papers/generic-parallel-functional/



Naive: Work O(N^2), Span O(N)

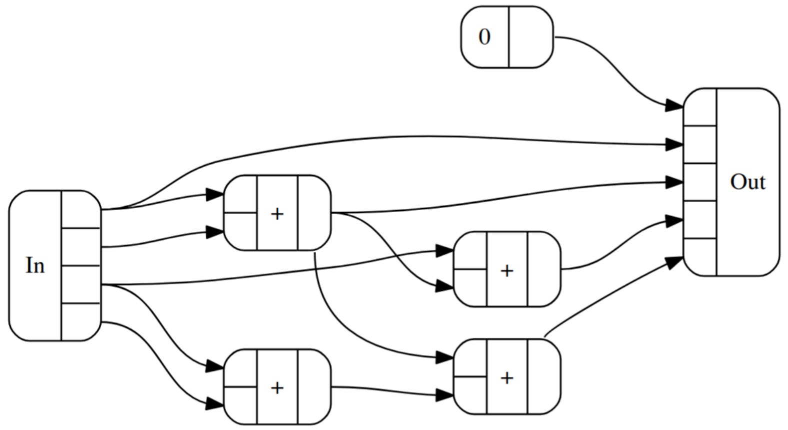


Fig. 18. *lscan* @(*Bush* 1) *[W=4, D=2]*

Conal Elliott. 2017. Generic functional parallel programming: Scan and FFT.

Optimal: Work O(N), Span O(log(N))

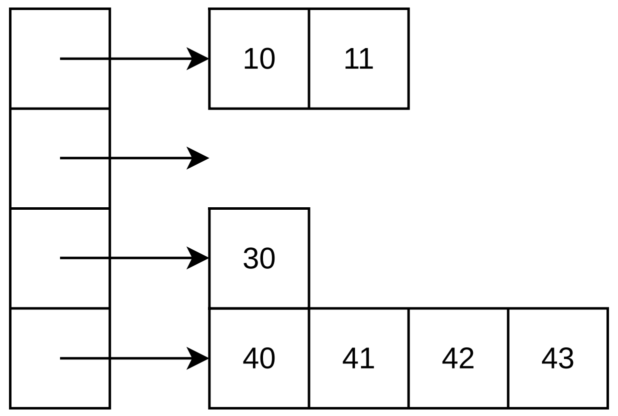
Parallel Scan in Futhark

```
def prefix_sum [n] (xs: [n]f64) : [n]f64 =
  scan (+) 0 xs
```

demo/scan.fut

demo/counting_sort.fut

Sparse Arrays



10	11	0	0
0	0	0	0
30	0	0	0
40	41	42	43

When to use them

- Graph algorithms
- One-hot encodings
- Hessian matrices
- Linear and quadratic programming
- Sparse voxel grids
- ...

Coordinate list (COO)

10	11	0	0
0	0	0	0
30	0	0	0
40	41	42	43

Rows: [0, 0, 2, 3, 3, 3, 3]

Columns: [0, 1, 0, 0, 1, 2, 3]

Values: [10, 11, 30, 40, 41, 42, 43]

Compressed Sparse Row (CSR)

10	11	0	0
0	0	0	0
30	0	0	0
40	41	42	43

Rows: [0, 2, 2, 3, 7]

Columns: [0, 1, 0, 0, 1, 2, 3]

Values: [10, 11, 30, 40, 41, 42, 43]

ELLPACK (ELL)

10	11	0	0
0	0	0	0
30	0	0	0
40	41	42	43

Columns:

[[0, 1, -, -], [-, -, -, -], [0, -, -, -], [0, 1, 2, 3]]

Values:

[[10, 11, 0, 0], [0, 0, 0, 0], [30, 0, 0, 0], [40, 41, 42, 43]]

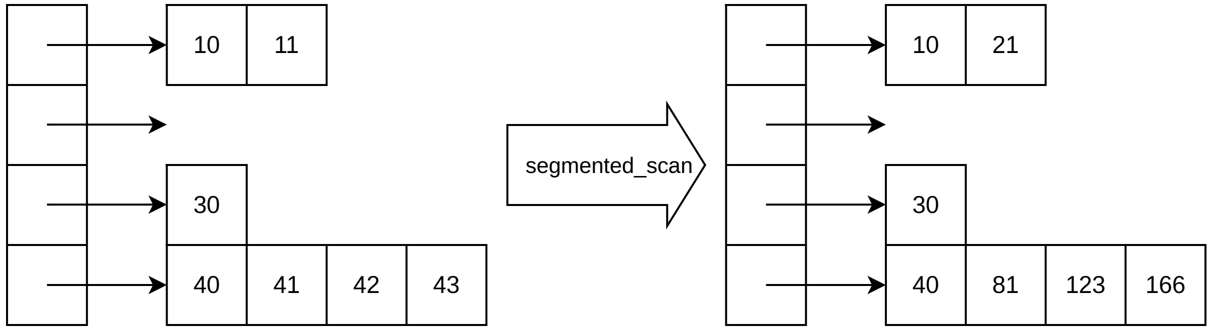
demo/coo.fut

demo/csr.fut

demo/ell.fut

Segmented Scan

Perform a scan operation on each array segment.



Example: row-wise summation.

Segmented Scan in Futhark

```
def segmented_scan_add [k] (flags: [k]bool) (values: [k]f64): [k]f64 =
  let combine (b1, v1) (b2, v2) = (b1 || b2, if b2 then v2 else v1 + v2)
  let pairs = scan combine (false, 0.0) (zip flags values)
  in map (.1) pairs
```

demo/segmented.fut

Summary and Outlook

Nested data parallelism is hard.

Next week: task parallelism.