Co-dfns Report

Dyalog ‘22

Aaron Hsu - aaron@dyalog.com

Why Co-dfns?

Make APL more scalable/accessible

Parallelism

Domains

How to APL

+

Where to APL

How?

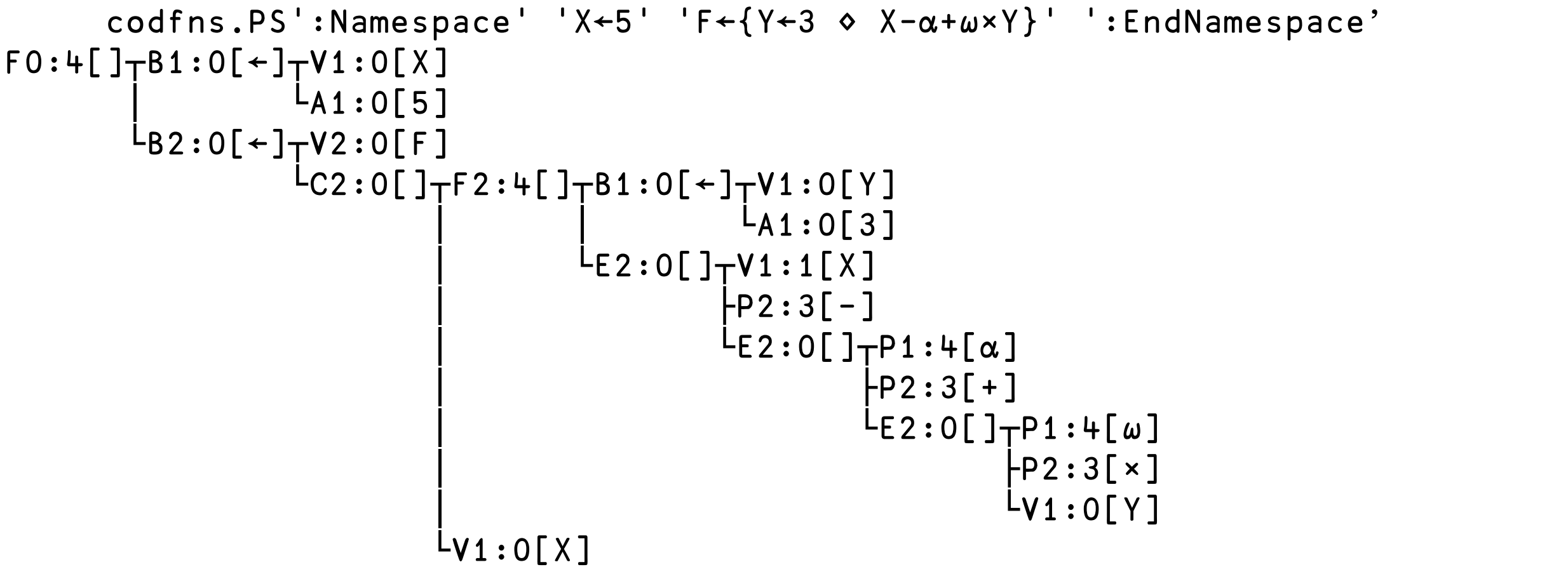
Use APL everywhere

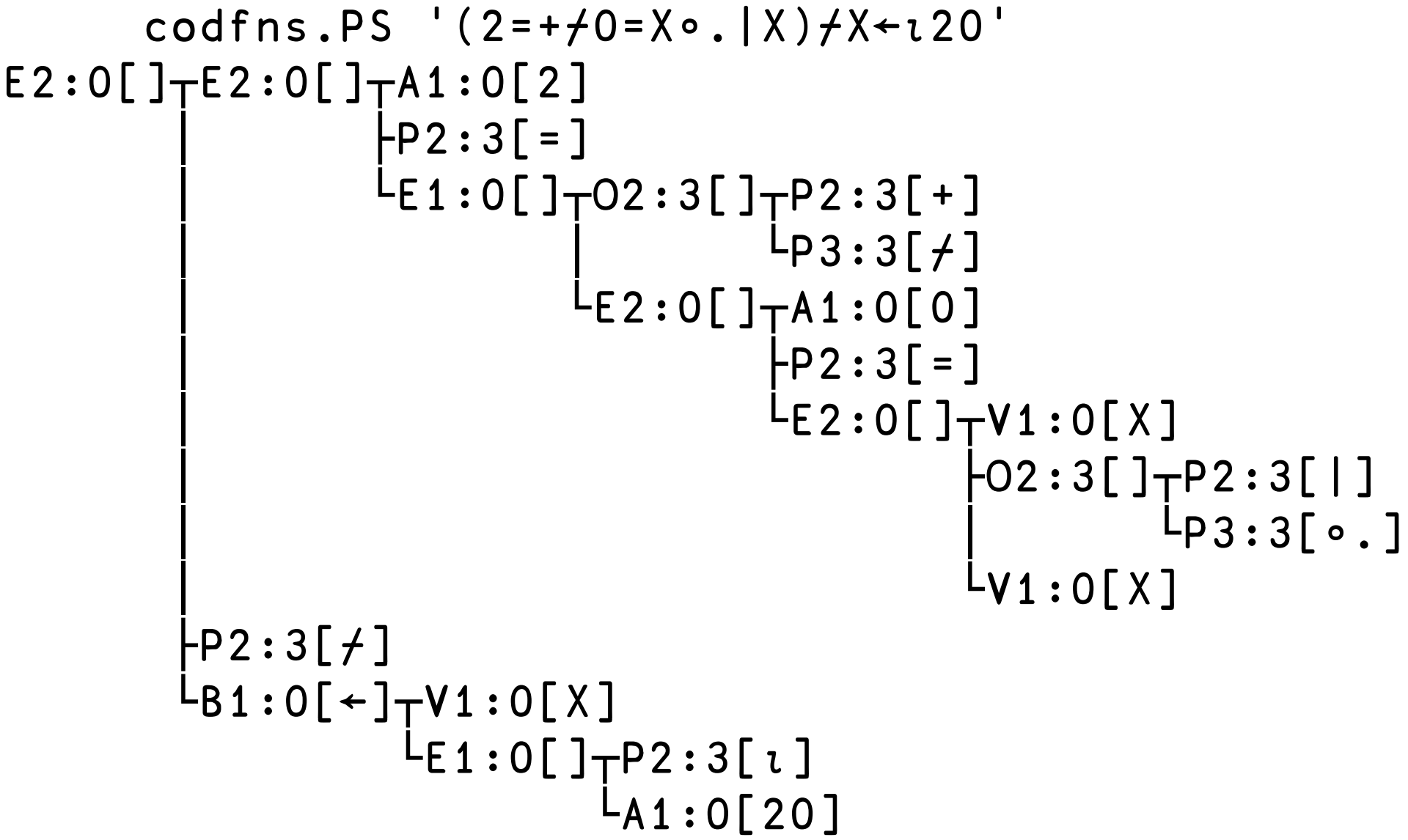
Data-parallel tree processing

Data-parallel Parser + Code-generation

Much faster (≥62×)
Error handling
Ambiguous APL
Type inference
Language features
Input types

Public Parser API


codfns.PS':Namespace' 'X←5' 'F←{Y←3 ⋄ X-α+ω×Y}' ':EndNamespace'
F0:4[]
B1:0[←]
V1:0[X]
A1:0[5]
B2:0[←]
V2:0[F]
C2:0[]
F2:4[]
B1:0[←]
V1:0[Y]
A1:0[3]
E2:0[]
V1:1[X]
P2:3[-]
E2:0[]
P1:4[α]
P2:3[+]
E2:0[]
P1:4[ω]
P2:3[×]
V1:0[Y]
V1:0[X]


codfns.PS '(2=+⌿0=X∘.|X)⌿X←⍳20'
E2:0[]
E2:0[]
A1:0[2]
P2:3[=]
E1:0[]
O2:3[]
P2:3[+]
P3:3[⌿]
E2:0[]
A1:0[0]
P2:3[=]
E2:0[]
V1:0[X]
O2:3[]
P2:3[|]
P3:3[∘.]
V1:0[X]
P2:3[⌿]
B1:0[←]
V1:0[X]
E1:0[]
P2:3[⍳]
A1:0[20]

LBL (dwh pp3) p
LBL (dwv pp3) p

(d t k n lx pos end)(xn xt)sym IN

```
      codfns.PS':Namespace' 'F←{(α+ω}' ':EndNamespace'
SYNTAX ERROR: MISMATCHED PARENS/BRACKETS
[2] F←{(α+ω}

      codfns.PS ':Namespace' 'F←{(αα ω)[)]}' ':EndNamespa'
SYNTAX ERROR: UNRECOGNIZED KEYWORD(S)
[3] :EndNamespa

      codfns.PS ':Namespace' 'F←{[αα ω]}' ':EndNamespace'
SYNTAX ERROR: BRACKET SYNTAX REQUIRES FUNCTION OR ARRAY TO ITS LEFT
[2] F←{[αα ω]}
```

Static Analysis

“Text Processing in APL”

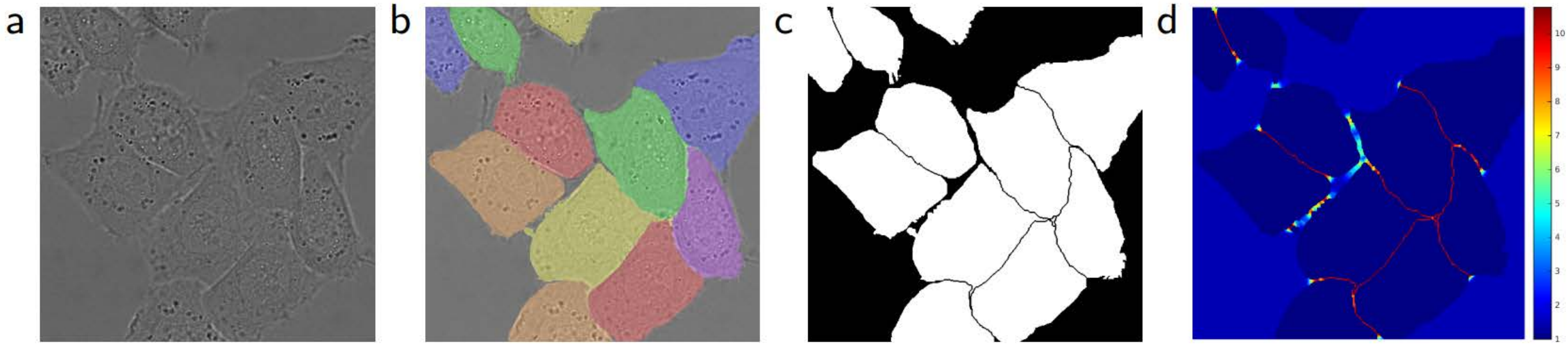
Data-parallel code gen?

Where?

GPUs

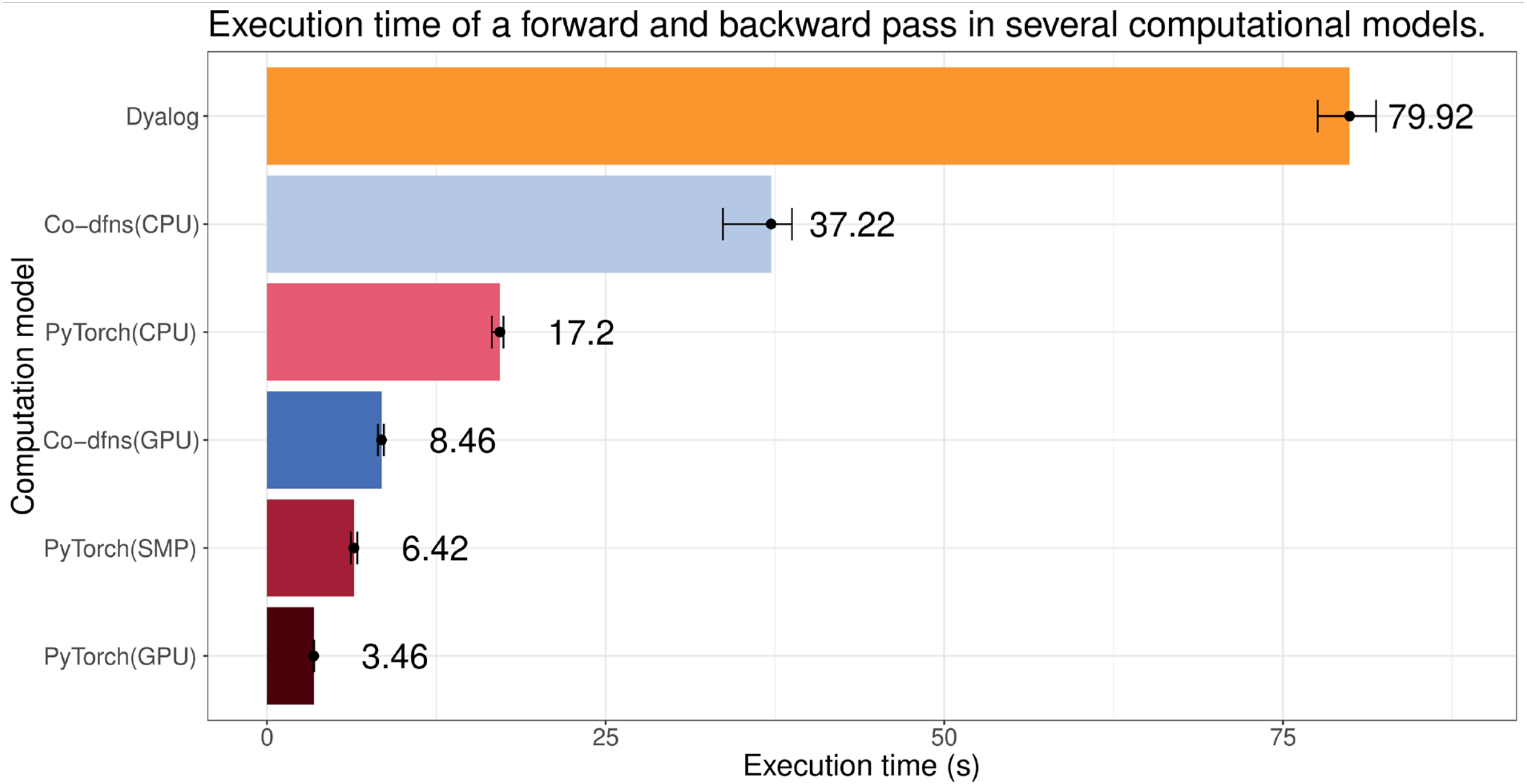
ML + HPC

U-net
Neural Network



(Ronneberger, Fischer, and Brox 2015)

0 2 1 3 4φω+.×α


Execution time of a forward and backward pass in several computational models.
Computation model
Dyalog
Co−dfns(CPU)
PyTorch(CPU)
Co−dfns(GPU)
PyTorch(SMP)
PyTorch(GPU)
79.92
37.22
17.2
8.46
6.42
3.46
0
25
50
75
Execution time (s)

New platforms

=

New applications

Portable Code Generation

Separate runtime distributable

Runtime implemented in APL

Retargeting
code gen + kernel
retargets the runtime

Flexibility

Extensibility

Low-overhead foreign function interface

‘foreign’I

First-class namespace objects

Integrate with other software environments

Javascript/WASM
Python
Java/C#
Scheme/Lisp
Custom Hardware

Native Interface + DWA

```
int
my_func(ARR **, ARR *, ARR *);

int
my_func_dwa(DWA *, DWA *, DWA*);
```

Optimized Runtimes

CPU vectors
Energy efficiency
Space efficiency
Scalar-optimized

CPU + GPU
Data buffers

What to expect...

New runtime may run slower initially

Fuller coverage of APL primitives

Characters
Mixed Arrays
Complex Numbers
Noble Arrays+
Nesting

Add the big “missing language features”

Trad-fns
Dynamic Scope
Control Structures
Branching
Error trapping/guards
Selective assignment
Execute/ambiguous values
Exporting user-defined operators
Namespaces

Lexical/Dynamic Scoping "done right"

Closures

Tracing and Debugging

Begin retargeting to new platforms

Publish/document new innovations and techniques

New work is driven by project interest, especially code.

Please send code!

Broken or working

Thank you.