

# Bend vs Mojo

Гурбанов Михаил



# Михаил Гурбанов



- работаю в Райфе
- старший разработчик
- развиваю сообщество
- выступаю в разных местах



Питону нужна замена?

# Scientific computing



- CPU/GPU bound
  - DS/ML - много матриц
  - Криптография - шифровка и дешифровка
  - Численные методы - еще больше матриц
  - Сжатие и распаковка

# Параллелизм



```
import multiprocessing

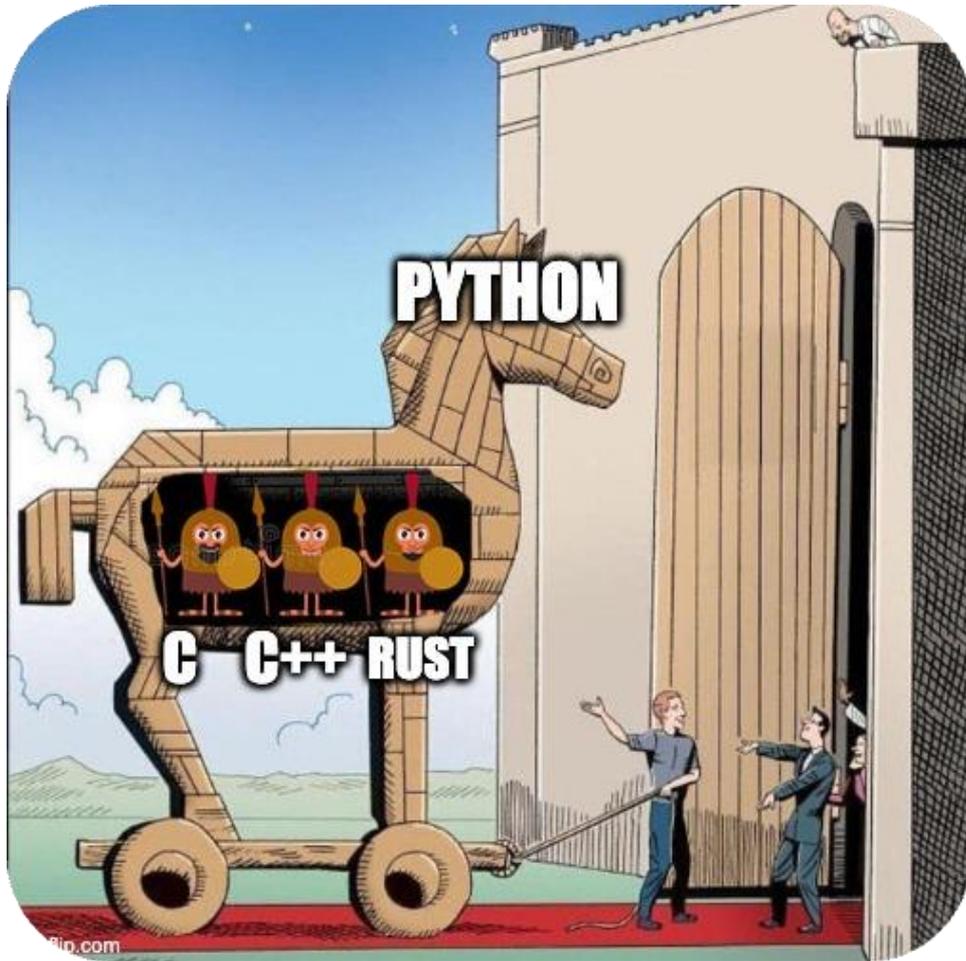
def main():
    numbers = [1, 2, 3, 4, 5]

    process = multiprocessing.Process(
        target=square_numbers, args=(numbers,)
    )

    process.start()

    process.join()
```

- В DS не засунешь
- Тяжелый код
- Python медленный



# Как еще можно ускорить

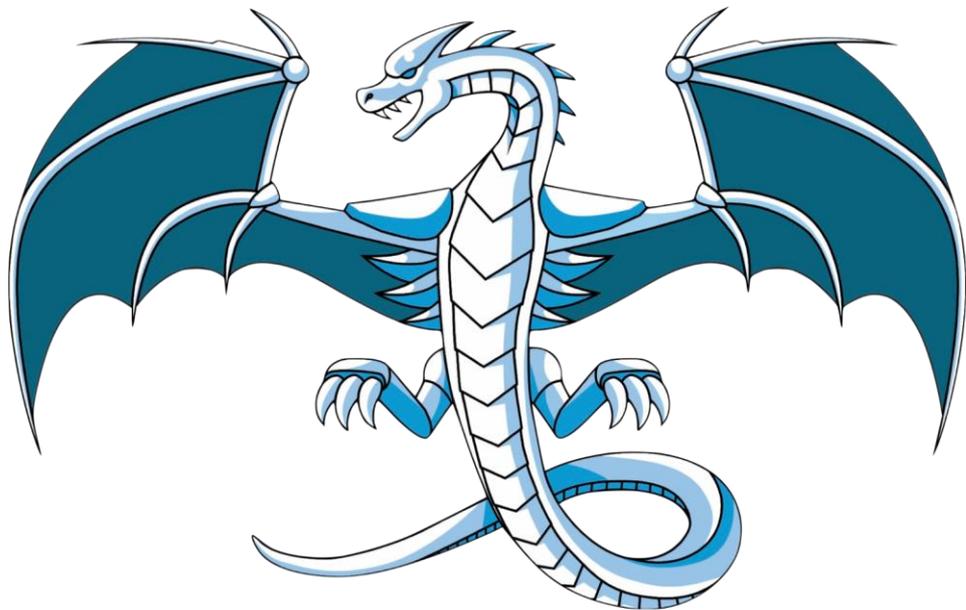


- Купить еще больше железа
- ~~Помолиться Гвидо~~ Ускорить CPython
- Использовать PyPy, Cython или другие
- Использовать либы на C/C++/Rust

# Chris Lattner



# Chris Lattner



# Chris Lattner



# Chris Lattner



# Modular



- Туса с ИИ платформой
- Максимальный performance
- Должно быть “хорошо”
- Не хватило C/C++/Rust





Mojo 



# Mojo



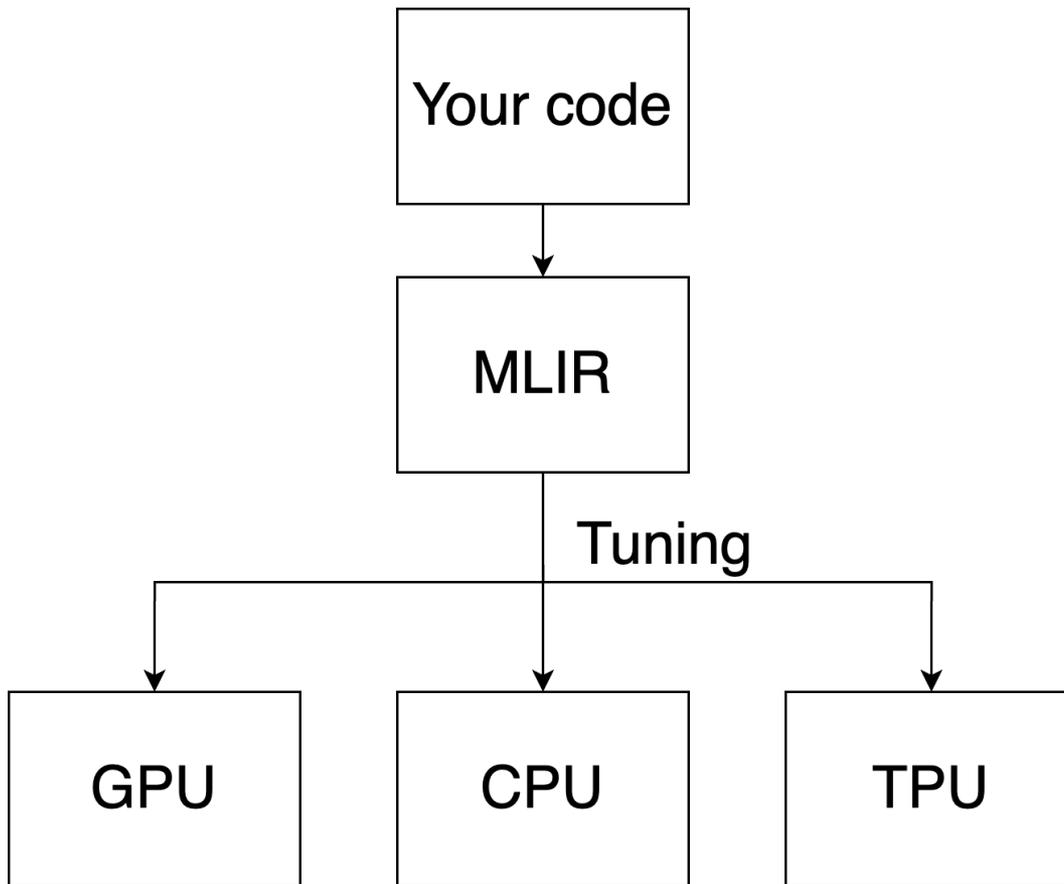
LANGUAGES	TIME (S)*	SPEEDUP VS PYTHON
Python 3.10.9	1027s	1X
PYPY	46.1s	22x
Scalar C++	0.20s	5,000x
Mojo🔥	0.03s	68,000x

# MLIR



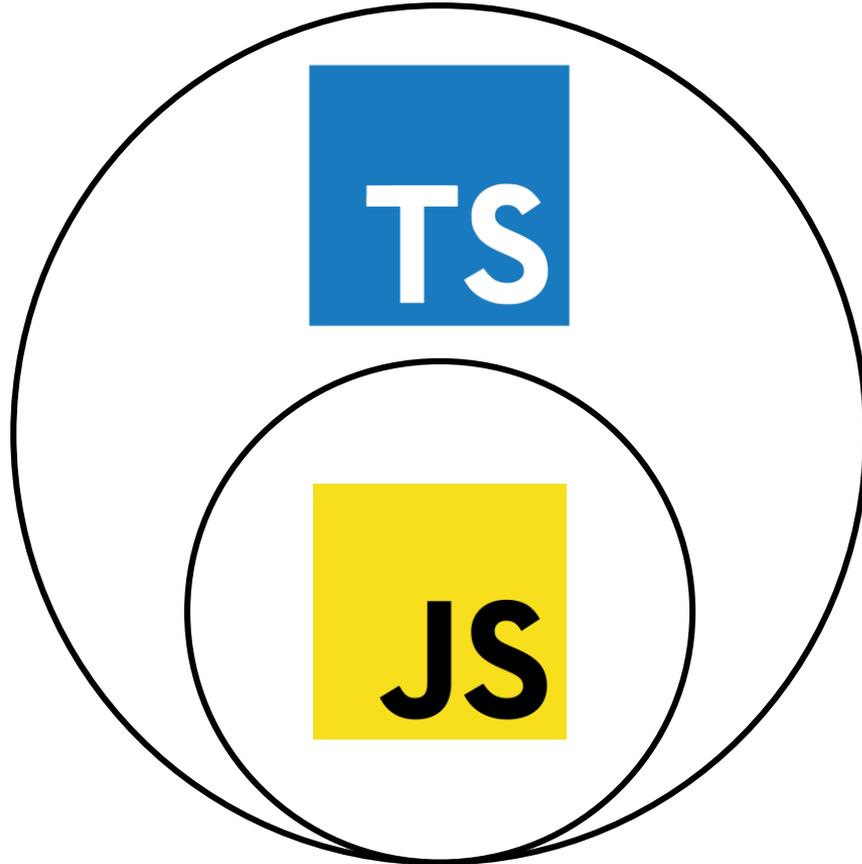
- Multi Level Intermediate Representation
- Фреймворк для создания оптимизационных компиляторов
- Можно компилировать код под любое железо

# MLIR





# Mojo



# Mojo



- Не совсем superset
- Вообще другой язык
- Компилируемый язык
- Статическая типизация
- Пока не совсем opensource



# Mojo: Типы данных



Простые типы	Сложные типы
Int8..64	List
UInt8..64	Dict
Float16..64	Set
String	Optional
Boolean	
SIMD	

# Mojo: Переменные



```
var number = 42
var string: String = "Mojo"

number = "some-string" # Error
```

# Mojo: Переменные



```
number = 42  
string = "Mojo"  
  
number = "some-string" # Error
```

# Мојо: Функции



```
def greet(name):  
    greeting = "Hello, " + name + "!"  
    return greeting
```

# Мојо: Функции



```
fn greet(name: String) -> String:  
    greeting = "Hello, " + name + "!"  
    return greeting
```

# Mojo: Структуры



```
struct MyPair:  
    var first: Int  
    var second: Int  
  
    fn __init__(inout self, first: Int, second: Int):  
        self.first = first  
        self.second = second
```

# Мојо: Ошибки



```
fn raise_an_error() raises:  
    raise Error("I'm an error!")  
  
fn bar():  
    try:  
        raise_an_error():  
    except e:  
        ...
```

# Мојо: использование Python

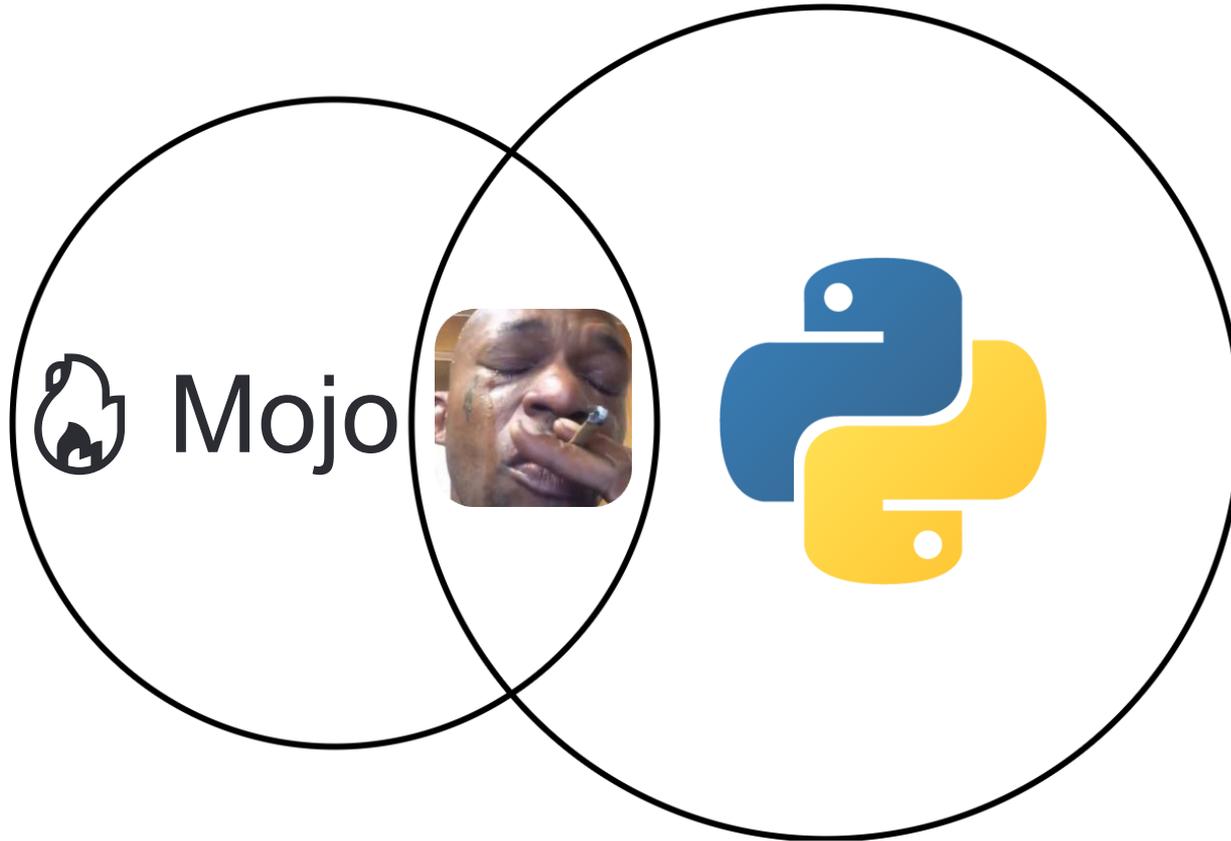


```
from python import Python

fn use_array() raises:
    var np = Python.import_module("numpy")

    var array = np.array([1, 2, 3])
    print(array)
```

# Mojo



# Мојо: Бенчи



- Mac M1
- `numpy.dot`
- Мојо

$$C_{ij} = \sum_{n=1}^k A_{in} * B_{nj}$$

# Мојо: Бенчи



```
struct Matrix[rows: Int, cols: Int]:  
  ...  
  
fn matmul_naive(C: Matrix, A: Matrix, B: Matrix):  
  for m in range(C.rows):  
    for k in range(A.cols):  
      for n in range(C.cols):  
        C[m, n] += A[m, k] * B[k, n]
```



# Мојо: Бенчи



```
struct Matrix[rows: Int, cols: Int]:  
  ...  
  
fn matmul_naive(C: Matrix, A: Matrix, B: Matrix):  
  for m in range(C.rows):  
    for k in range(A.cols):  
      for n in range(C.cols):  
        C[m, n] += A[m, k] * B[k, n]
```



 Мојо в 50 раз медленнее NumPy

# Мојо: Бенчи



```
from algorithm import vectorize

fn matmul_vectorized_1(C: Matrix, A: Matrix, B: Matrix):
  for m in range(C.rows):
    for k in range(A.cols):
      @parameter
      fn dot[nelts: Int](n: Int):
        C.store(
          m, n,
          C.load[nelts](m, n) + A[m, k] * B.load[nelts](k, n)
        )
      vectorize[dot, nelts, size = C.cols]()
```



# Мојо: Бенчи



```
from algorithm import vectorize

fn matmul_vectorized_1(C: Matrix, A: Matrix, B: Matrix):
    for m in range(C.rows):
        for k in range(A.cols):
            @parameter
            fn dot[nelts: Int](n: Int):
                C.store(
                    m, n,
                    C.load[nelts](m, n) + A[m, k] * B.load[nelts](k, n)
                )
            vectorize[dot, nelts, size = C.cols]()
```



 Мојо в 10 раз медленнее NumPy

# Мојо: Бенчи



```
from algorithm import parallelize

fn matmul_parallelized(C: Matrix, A: Matrix, B: Matrix):
    @parameter
    fn calc_row(m: Int):
        for k in range(A.cols):
            @parameter
            fn dot[nelts : Int](n : Int):
                C.store[nelts](
                    m, n,
                    C.load[nelts](m,n) + A[m,k] * B.load[nelts](k,n)
                )
            vectorize[dot, nelts, size = C.cols]()
        parallelize[calc_row](C.rows, C.rows)
```



# Мојо: Бенчи



```
from algorithm import parallelize

fn matmul_parallelized(C: Matrix, A: Matrix, B: Matrix):
    @parameter
    fn calc_row(m: Int):
        for k in range(A.cols):
            @parameter
            fn dot[nelts : Int](n : Int):
                C.store[nelts](
                    m, n,
                    C.load[nelts](m,n) + A[m,k] * B.load[nelts](k,n)
                )
            vectorize[dot, nelts, size = C.cols]()
    parallelize[calc_row](C.rows, C.rows)
```



 Мојо в 2.5 раз медленнее NumPy

# Мојо: Бенчи



```
fn matmul_unrolled_parallelized(C: Matrix, A: Matrix, B: Matrix):
  @parameter
  fn calc_row(m: Int):
    @parameter
    fn calc_tile[tile_x: Int, tile_y: Int](x: Int, y: Int):
      for k in range(y, y + tile_y):
        @parameter
        fn dot[nelts: Int](n: Int):
          C.store(
            m, n + x,
            C.load[nelts](m, n + x) + A[m, k] * B.load[nelts](k, n + x)
          )

        # Vectorize by nelts and unroll by tile_x/nelts
        # Here unroll factor is 4
        alias unroll_factor = tile_x // nelts
        vectorize[dot, nelts, size=tile_x, unroll_factor=unroll_factor]()

  parallelize[calc_row](C.rows, C.rows)
```



# Мојо: Бенчи



```
fn matmul_unrolled_parallelized(C: Matrix, A: Matrix, B: Matrix):
  @parameter
  fn calc_row(m: Int):
    @parameter
    fn calc_tile[tile_x: Int, tile_y: Int](x: Int, y: Int):
      for k in range(y, y + tile_y):
        @parameter
        fn dot[nelts: Int](n: Int):
          C.store(
            m, n + x,
            C.load[nelts](m, n + x) + A[m, k] * B.load[nelts](k, n + x)
          )

        # Vectorize by nelts and unroll by tile_x/nelts
        # Here unroll factor is 4
        alias unroll_factor = tile_x // nelts
        vectorize[dot, nelts, size=tile_x, unroll_factor=unroll_factor]()

    parallelize[calc_row](C.rows, C.rows)
```



Мојо в 87 раз быстрее numpy

# Мојо: Бенчи



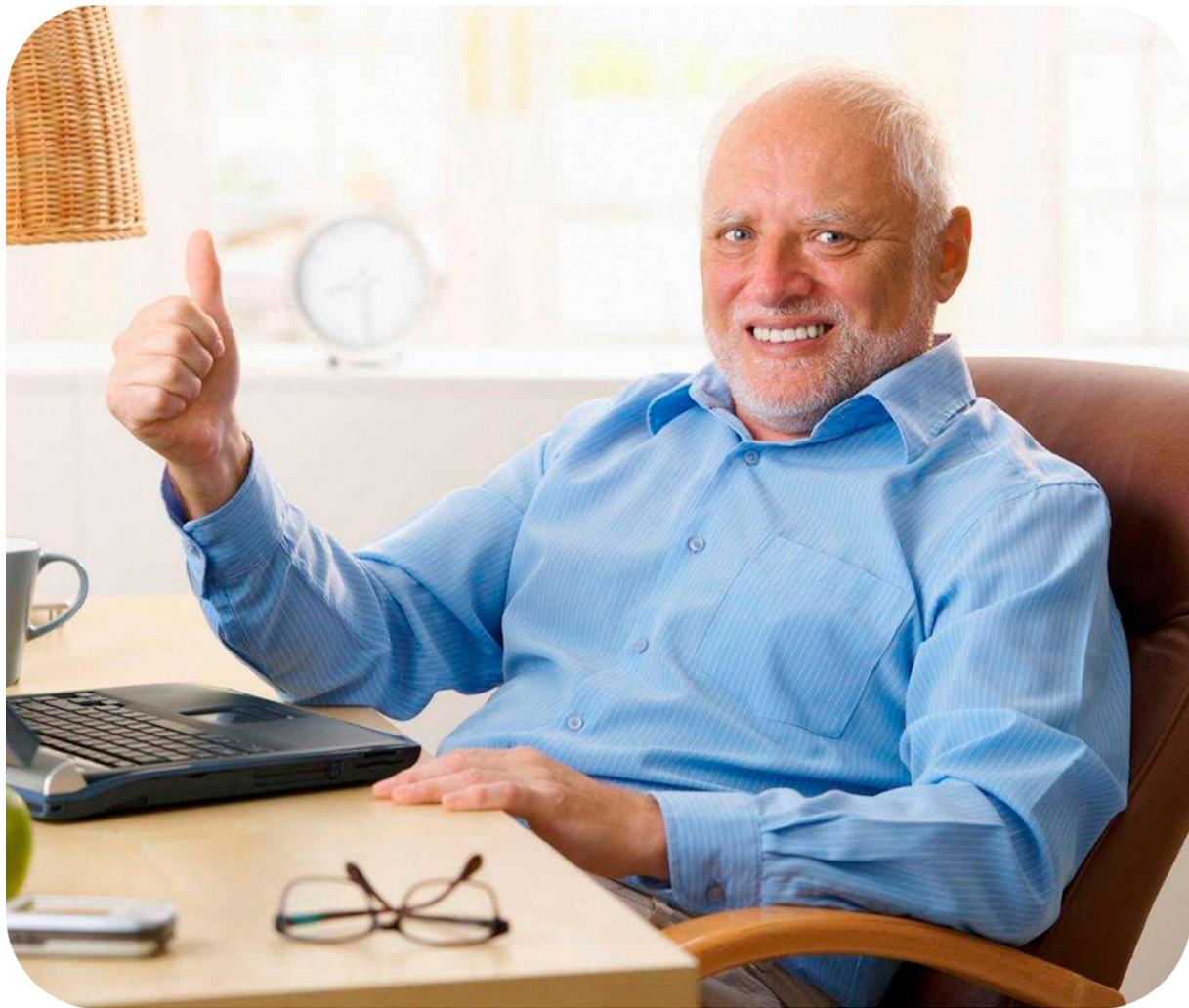
```
fn matmul_unrolled_parallelized(C: Matrix, A: Matrix, B: Matrix):
    @parameter
    fn calc_row(m: Int):
        @parameter
        fn calc_tile[tile_x: Int, tile_y: Int](x: Int, y: Int):
            for k in range(y, y + tile_y):
                @parameter
                fn dot[nelts: Int](n: Int):
                    C.store(
                        m, n + x,
                        C.load[nelts](m, n + x) + A[m, k] * B.load[nelts](k, n + x)
                    )

                # Vectorize by nelts and unroll by tile_x/nelts
                # Here unroll factor is 4
                alias unroll_factor = tile_x // nelts
                vectorize[dot, nelts, size=tile_x, unroll_factor=unroll_factor]()

    parallelize[calc_row](C.rows, C.rows)
```



Мојо в 7 раз быстрее jax + jit



# Моjo: Эко-система



awesome-mojo Public Watch 28 Fork 48 Star 812

main 1 Branch 0 Tags  Add file Code

**automata** Merge pull request #23 from ChromiteExabyte/ChromiteExabyte-URL-re... 4a9c69c · 2 weeks ago 63 Commits

LICENSE	Initial commit	last year
README.md	Update README.md	2 weeks ago
code-of-conduct.md	Add contributing guide	last year
contributing.md	Add contributing guide	last year

[README](#) [Code of conduct](#) [CC0-1.0 license](#) ⋮

## Awesome Mojo

A curated list of awesome Mojo frameworks, libraries, software and resources.

If you want to contribute, please read [this guide](#).

### About

A curated list of awesome Mojo frameworks, libraries, software and resources

- Readme
- CC0-1.0 license
- Code of conduct
- Activity
- Custom properties
- 812 stars
- 28 watching
- 48 forks
- Report repository

### Releases

No releases published

# Мојо: Сообщество



 @modular\_ai

 company/modular\_ai

 @modularinc

 modular

# Mojo: Установка



```
$ curl -ssL https://magic.modular.com/069ba1af-3b4f-488b-b881-5566868898ef | bash
```

```
$ magic init my-project
```

```
$ magic run mojo --version
```

# Mojo: Установка



```
$ curl -ssL https://magic.modular.com/069ba1af-3b4f-488b-b881-5566868898ef | bash
```

```
$ magic init my-project
```

```
$ magic run mojo --version
```

```
$ magic add max
```



# MiAX

A new framework for Gen AI, and the  
best way to deploy PyTorch

SERVING LIBRARY

INFERENCE RUNTIME

MODEL PIPELINE

CLI

EXTEND EASILY IN



aws



SCALE IN THE CLOUD

# The best way to deploy PyTorch

Avg. Latency improvement

 + MAX

3.4x

 + MAX

2.7x

## SOTA performance in just 3 lines of code

Drop in your PyTorch or ONNX models and get an instant boost in performance with our next generation inference runtime for CPUs and GPUs

[See for yourself >](#)

\*CPU Benchmarks.

# Mojo: Max



```
from max import engine

# Load your model:
session = engine.InferenceSession()
model = session.load(model_path)

# Prepare the inputs, then run an inference:
outputs = model.execute(**inputs)

# Process the output here.
```

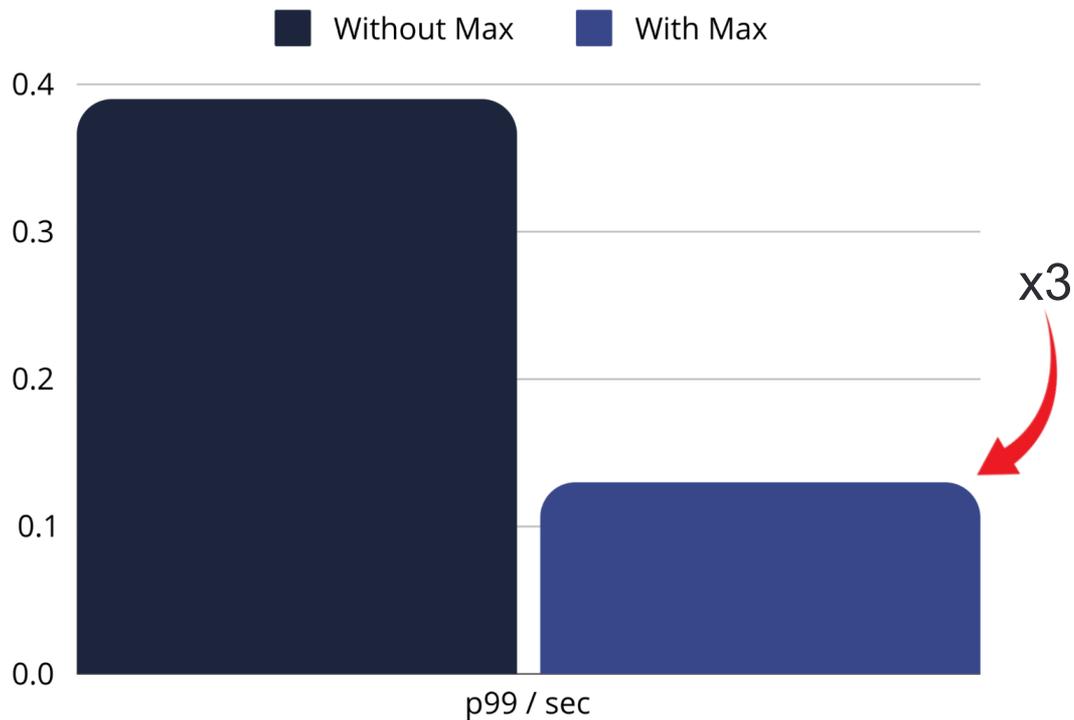
# Mojo: Max CPU benchmarks



- Mac M1
- torchscripted BERT
- Emotion dataset
- Python / Max engine



# Mojo: Max CPU benchmarks



# MAX on GPU waiting list



Be the first to get lightning fast inference speed on your GPUs. Be the envy of all your competitors and lower your compute spend.

First name\*

Last name\*

Email\*

Job title\*

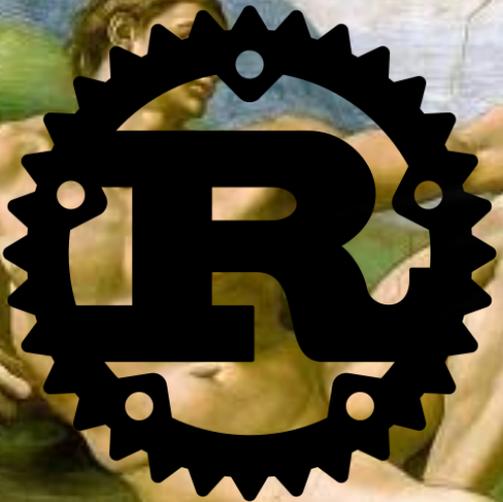
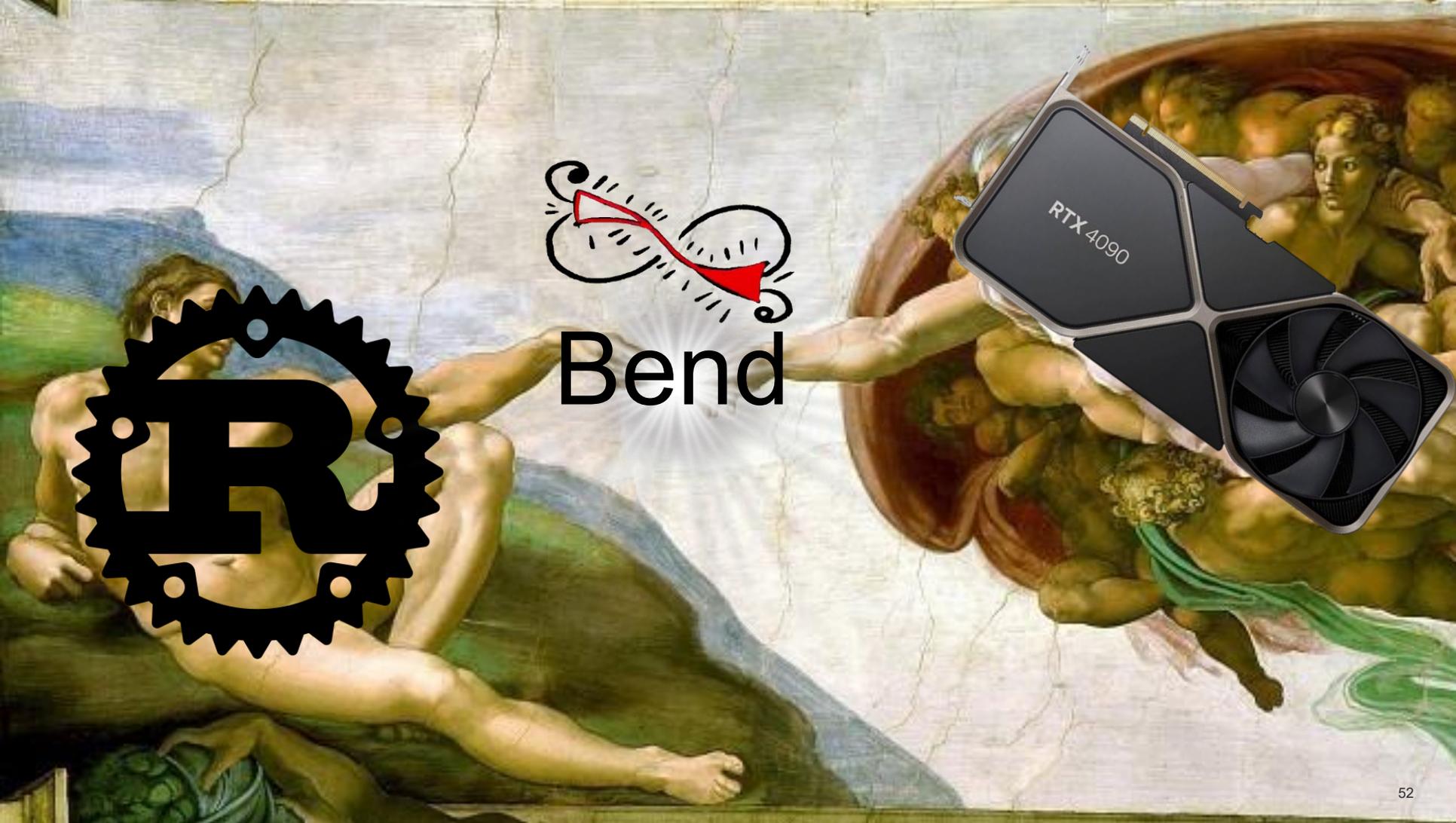


# Mojo: Вывод



- Намного лучше, чем раньше
- Развивающееся сообщество
- Хотят внедриться в ядро pytorch
- В прод бы пока не тащил
- MAX выглядит очень круто





Bend



# Bend



- Совсем молодой
- Massively Parallel
- In Rust ⚡
- Can parallel - will parallel
- CPU / GPU





## Interaction Combinators

Yves Lafont\*

*Institut de Mathématiques de Luminy, UPR 9016 du CNRS,  
163 avenue de Luminy, case 930, 13288 Marseille Cedex 9, France*

E-mail: [lafont@iml.univ-mrs.fr](mailto:lafont@iml.univ-mrs.fr)

---

It is shown that a very simple system of *interaction combinators*, with only three symbols and six rules, is a universal model of distributed computation, in a sense that will be made precise. This paper is the continuation of the author's work on *interaction nets*, inspired by Girard's proof nets for *linear logic*, but no preliminary knowledge of these topics is required for its reading. © 1997 Academic Press

---

Bend



420 + 69

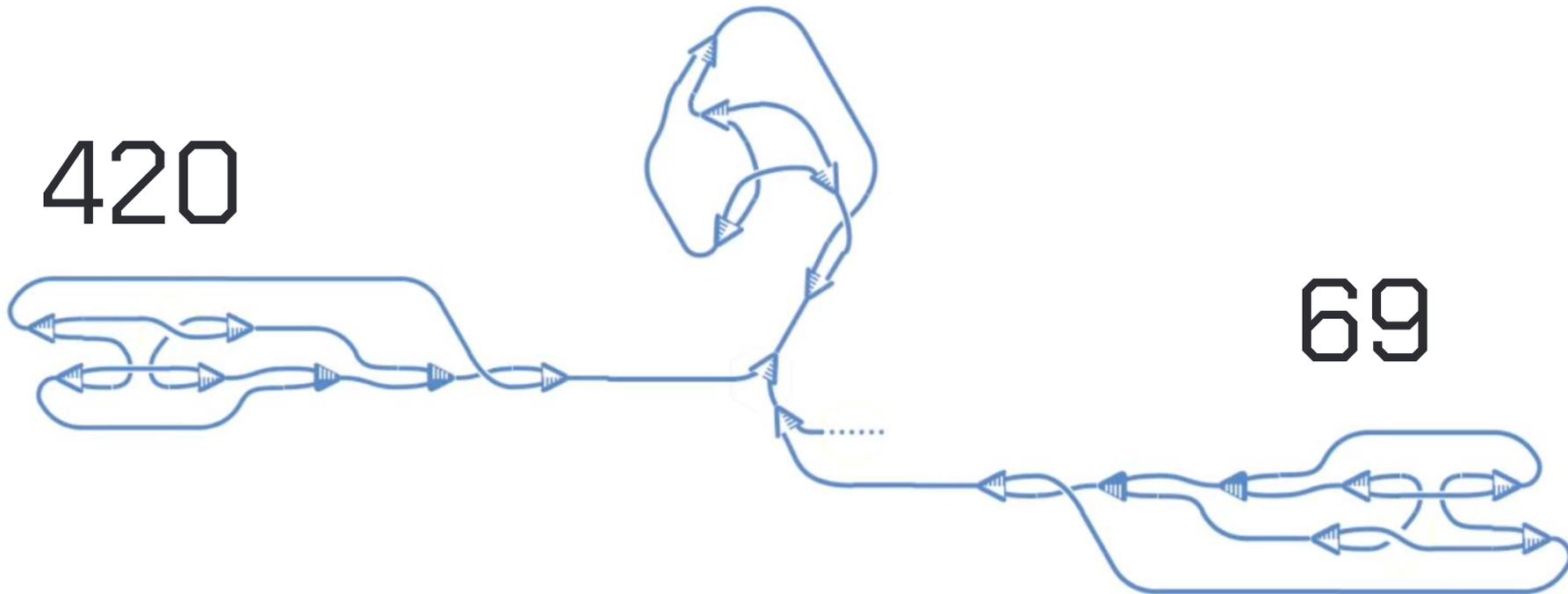
# Bend



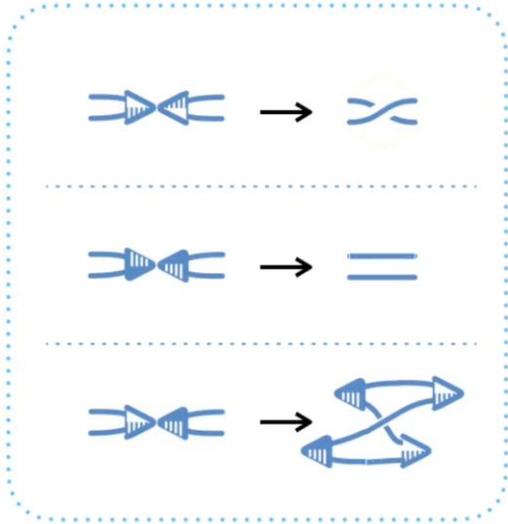
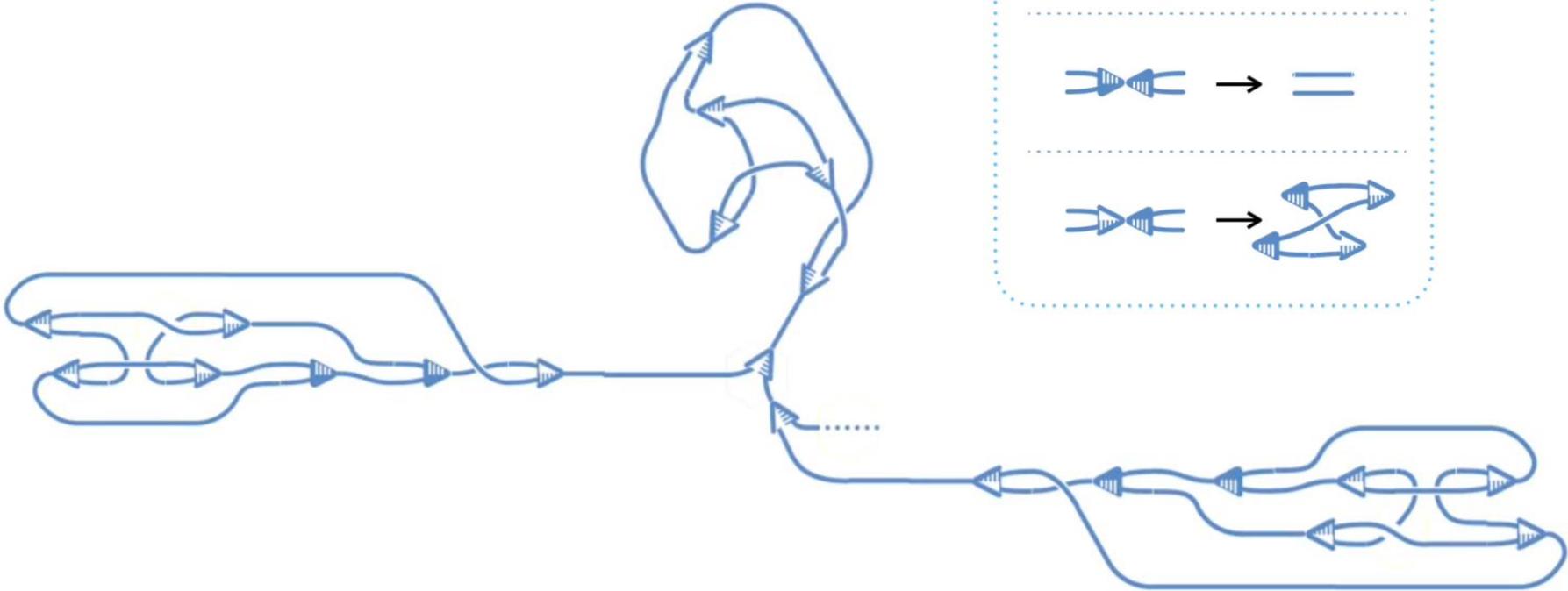
+

420

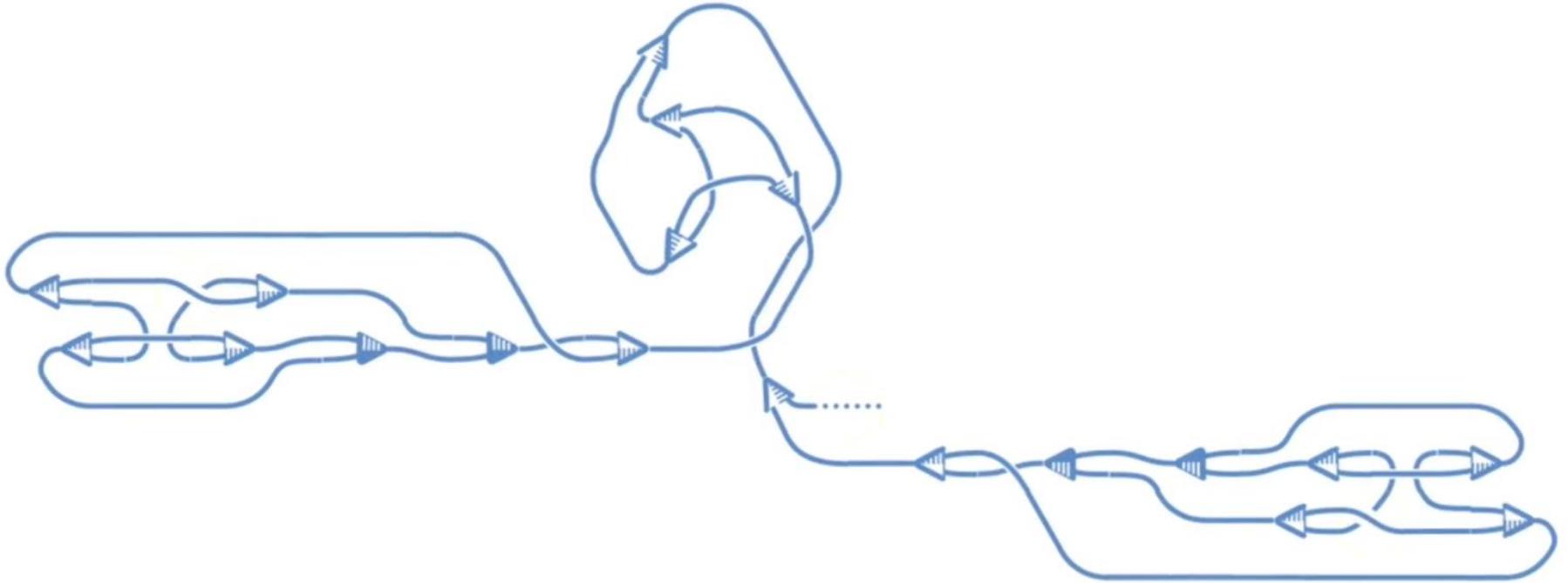
69



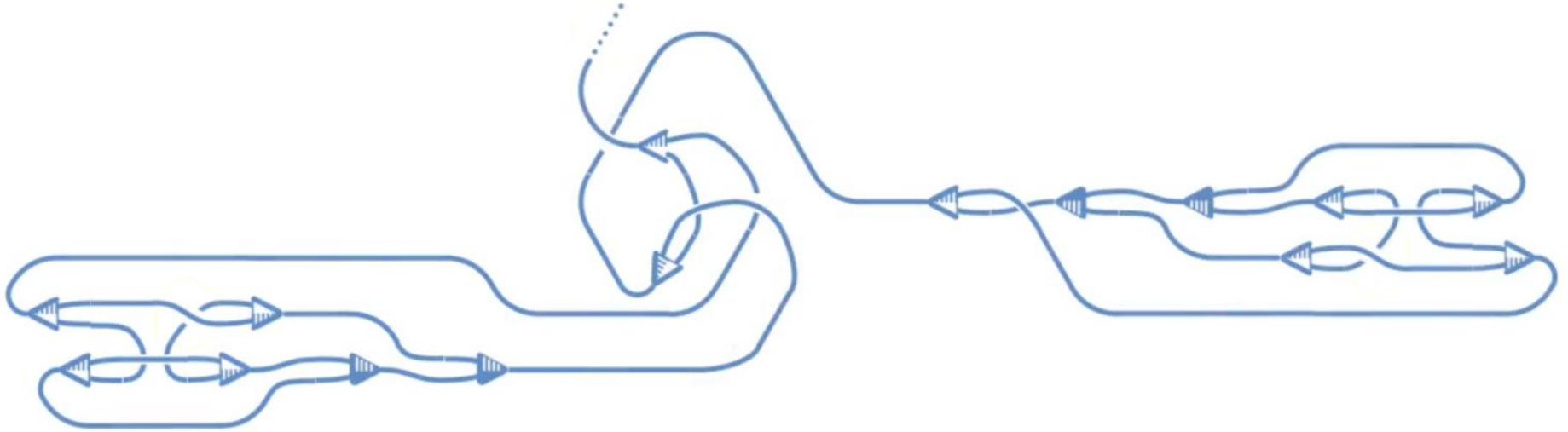
# Bend



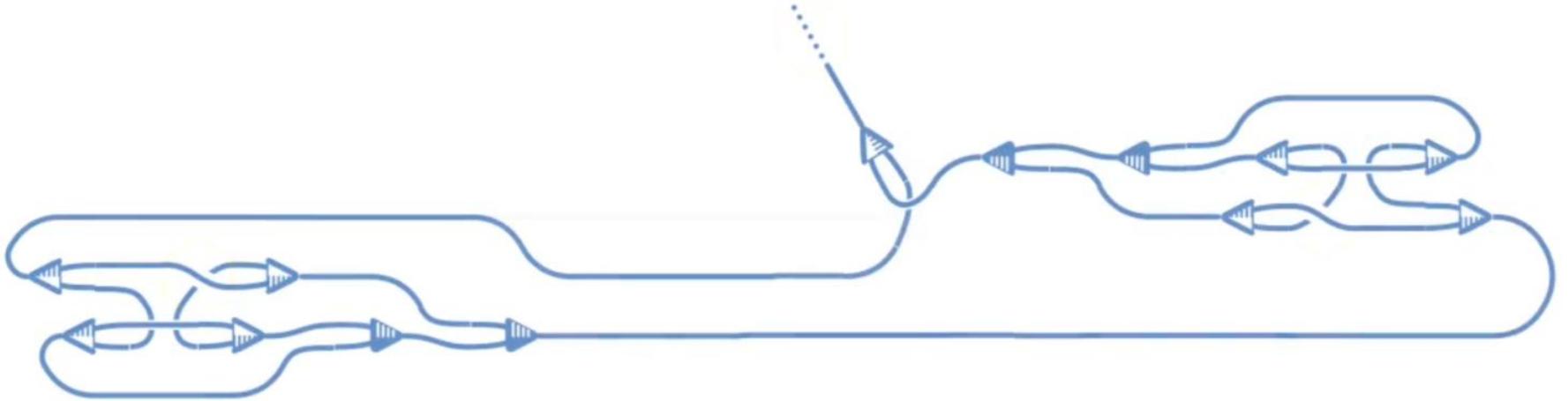
# Bend



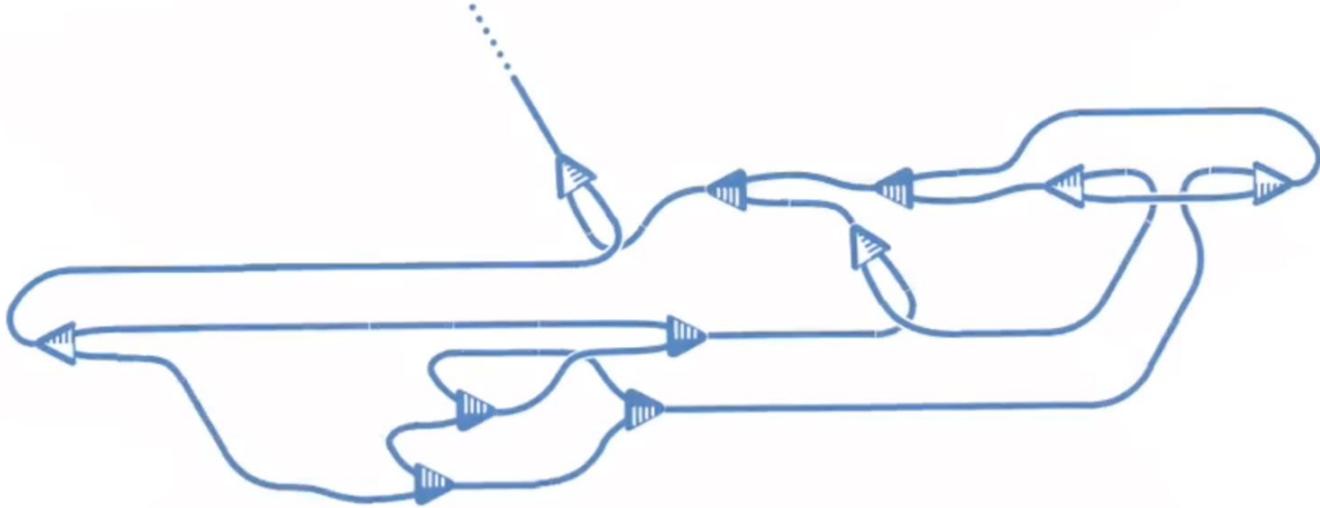
# Bend



# Bend



# Bend



Bend



$$420 + 69 = 489$$

# Bend: HVM



- Higher order Virtual Machine
- На основе Interaction Combinators
- Является параллельным рантаймом
- Для взаимодействия с ним и сделан Bend
- Можно обращаться из Bend напрямую

# Bend: HVM



```
@main = a
  & @sum ~ (28 (0 a))

@sum = (?(((a a) @sum__C0) b) b)

@sum__C0 = ({c a} ({$([*2] $([+1] d)) $([*2] $([+0] b))}) f))
  &! @sum ~ (a (b $([+] $(e f))))
  &! @sum ~ (c (d e))
```

# Венд: Функции



```
def distance(ax, ay, bx, by):  
    dx = bx - ax  
    dy = by - ay  
    return (dx * dx + dy * dy) ** 0.5  
  
def main():  
    return distance(10.0, 10.0, 20.0, 20.0)
```

# Vend: Объекты



```
object Point { x, y }
```

```
def distance(a, b):
```

```
  open Point: a
```

```
  open Point: b
```

```
  dx = b.x - a.x
```

```
  dy = b.y - a.y
```

```
  return (dx * dx + dy * dy) ** 0.5
```

```
def main():
```

```
  return distance(
```

```
    Point { x: 10.0, y: 10.0 },
```

```
    Point { x: 20.0, y: 20.0 }  
  )
```

# Bend: Типы



```
type Shape:
  Circle { radius }
  Rectangle { width, height }

def area(shape):
  match shape:
    case Shape/Circle:
      return 3.14 * shape.radius ** 2.0
    case Shape/Rectangle:
      return shape.width * shape.height

def main:
  return area(Shape/Circle { radius: 10.0 })
```

# Bend: Неизменяемость



```
def parity(x):  
    result = "odd"  
    if x % 2 == 0:  
        result = "even"  
    return result
```

# Bend: Неизменяемость

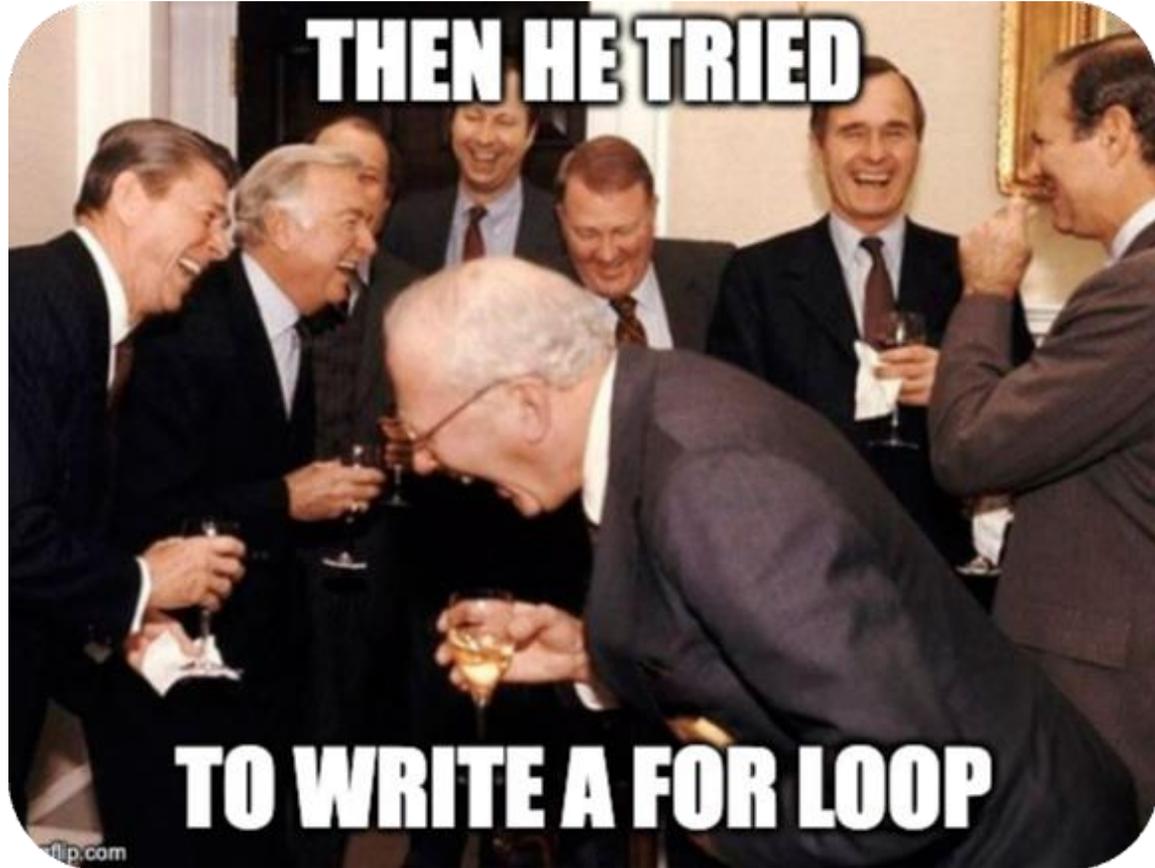


```
def partity(x):  
    if x % 2 == 0:  
        return "even"  
    else:  
        return "odd"
```

# Bend: Неизменяемость



```
def sequential_sum(number):  
    total = 0  
    for cur_index in range(number)  
        total += cur_index  
    return total
```



# Bend: List



```
type List:  
  Nil  
  Cons {head, ~tail}
```

# Bend: List



```
def main:  
  my_list = List/Cons {  
    head: 1, tail: List/Cons {  
      head: 2, tail: List/Cons {  
        head: 3, tail: List/Nil  
      }  
    }  
  }  
  return my_list
```

# Bend: List



```
def main:  
    my_list = [1, 2, 3]  
    return my_list
```

# Bend: Fold

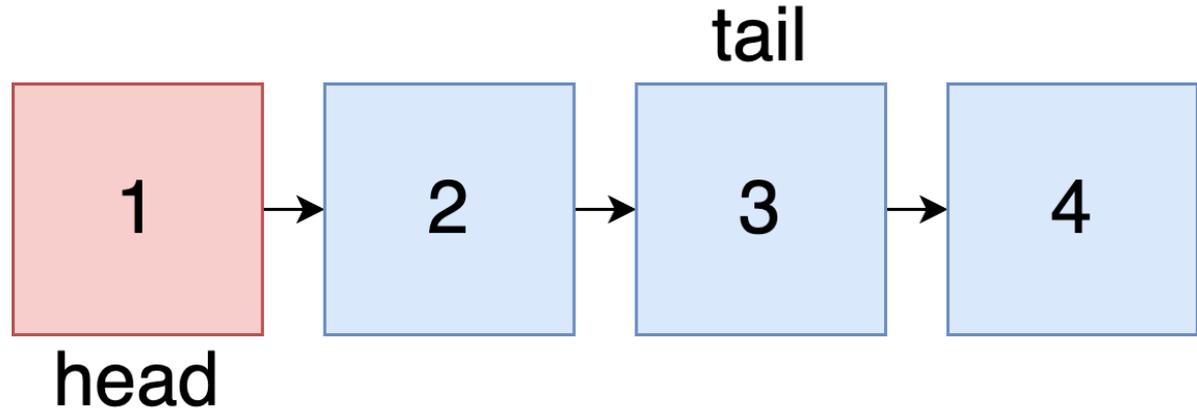


```
def list_sum(input_list):  
  fold input_list:  
    case List/Cons:  
      return input_list.head + input_list.tail  
    case List/Nil:  
      return 0
```

# Bend: Fold



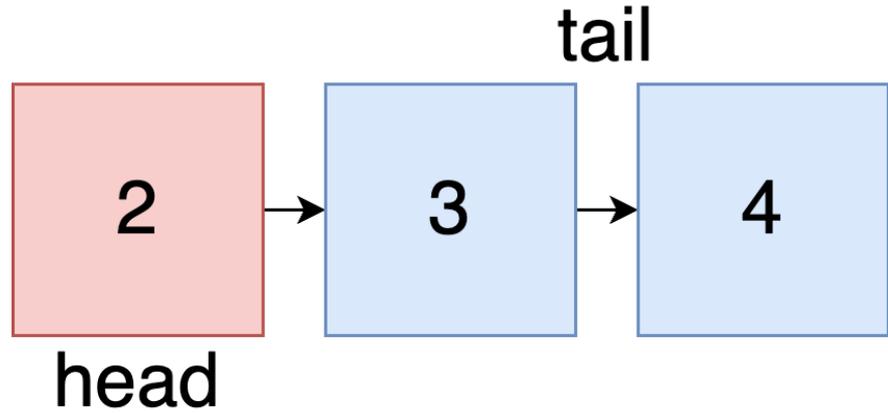
1) (1 + tail)



# Bend: Fold



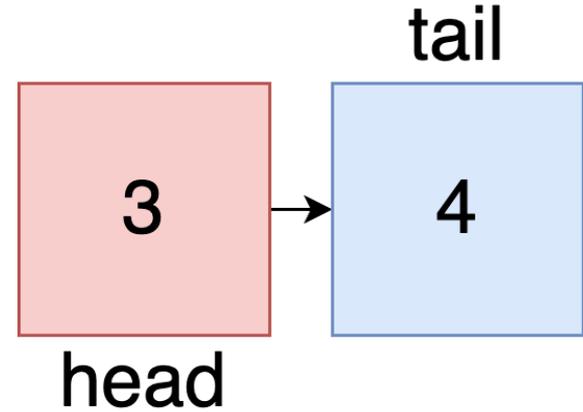
- 1)  $(1 + \text{tail})$
- 2)  $(1 + (2 + \text{tail}))$



# Bend: Fold



- 1)  $(1 + \text{tail})$
- 2)  $(1 + (2 + \text{tail}))$
- 3)  $(1 + (2 + (3 + \text{tail})))$



# Bend: Fold



- 1)  $(1 + \text{tail})$
- 2)  $(1 + (2 + \text{tail}))$
- 3)  $(1 + (2 + (3 + \text{tail})))$
- 4)  $(1 + (2 + (3 + (4 + 0))))$



head

# Bend: Fold



- 1) (1 + tail)
- 2) (1 + (2 + tail))
- 3) (1 + (2 + (3 + tail)))
- 4) (1 + (2 + (3 + (4 + 0))))



# Bend: Tree



```
type Tree:  
  Node {~left, ~right}  
  Leaf {value}  
  
def main:  
  tree = ![![!1, !2], ![!3, !4]]
```

# Bend: Fold

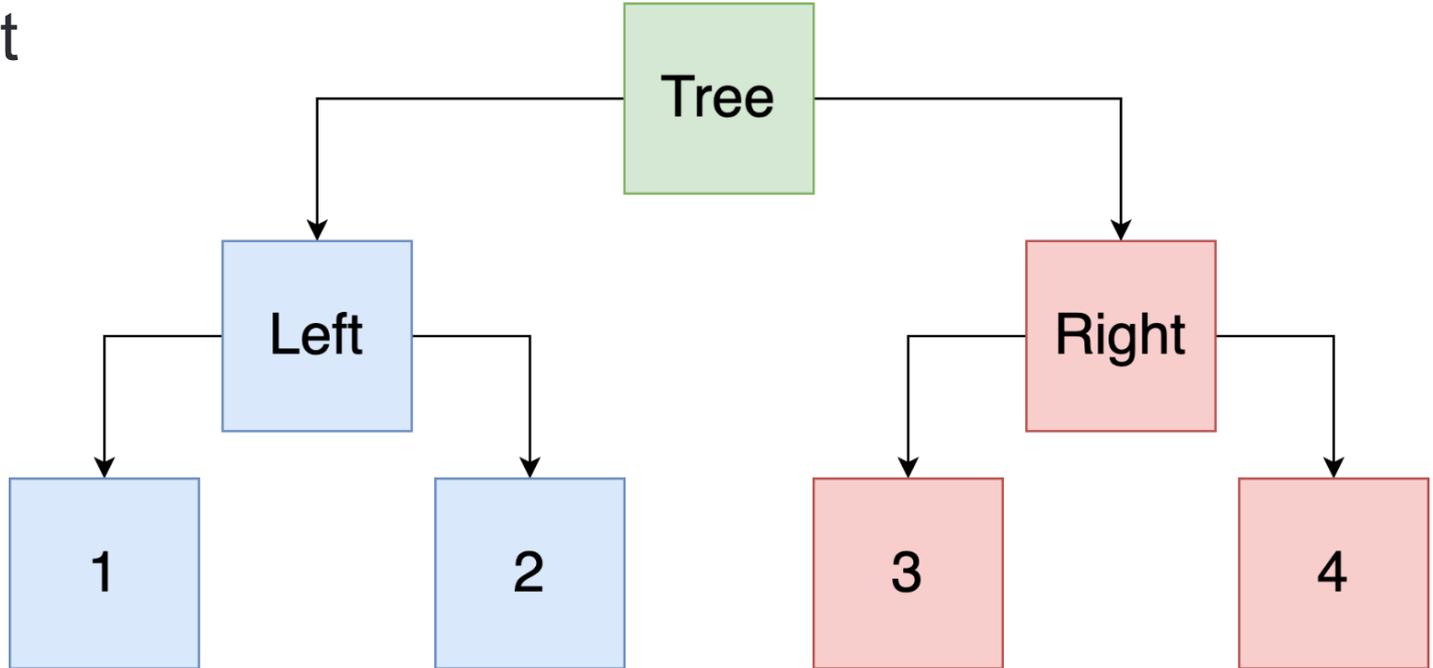


```
def tree_sum(tree):  
  fold tree:  
    case Tree/Node:  
      return tree.left + tree.right  
    case Tree/Leaf:  
      return tree.value
```

# Bend: Fold



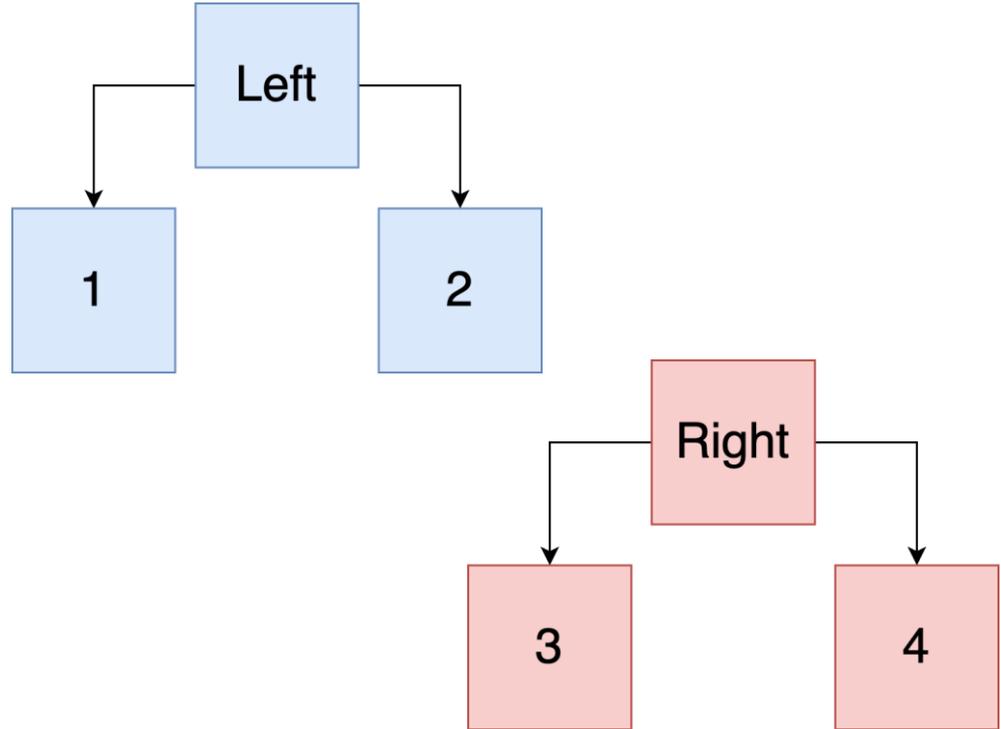
1) left + right



# Bend: Fold



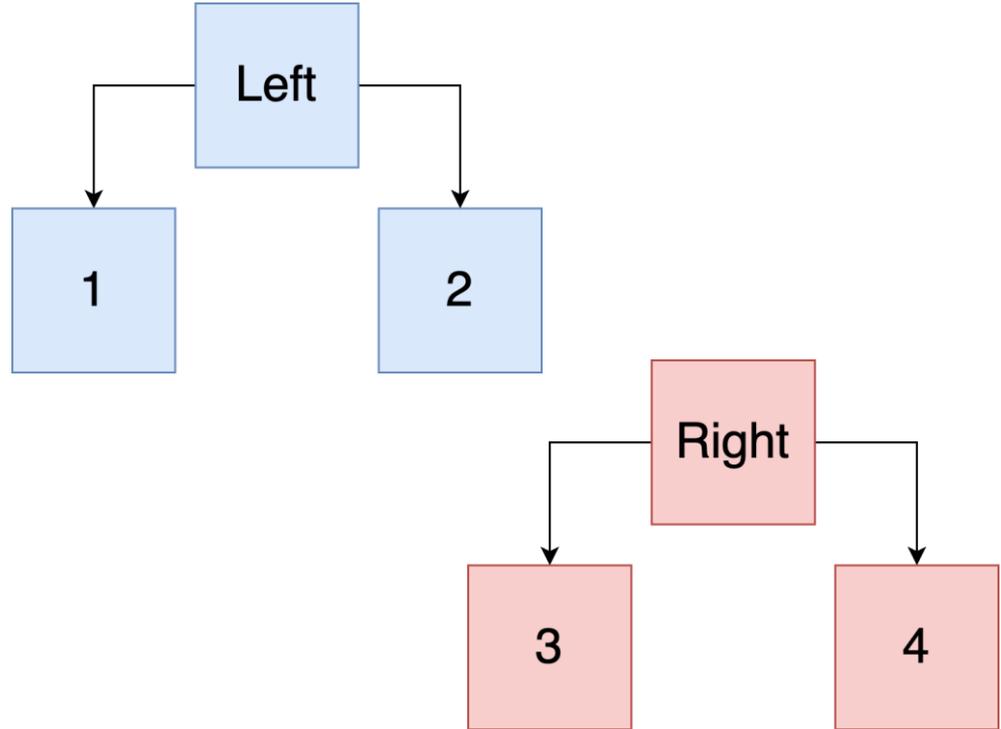
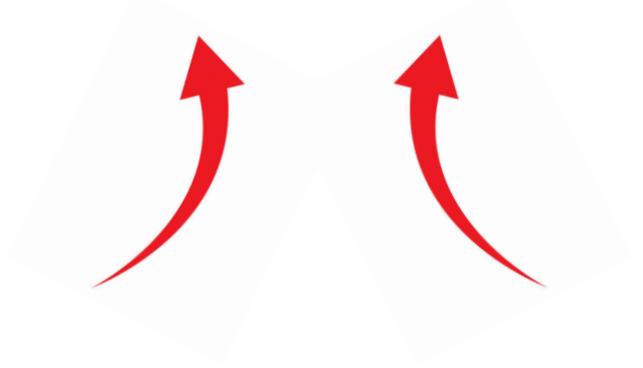
- 1) left + right
- 2)  $(1 + 2) + (3 + 4)$



# Bend: Fold



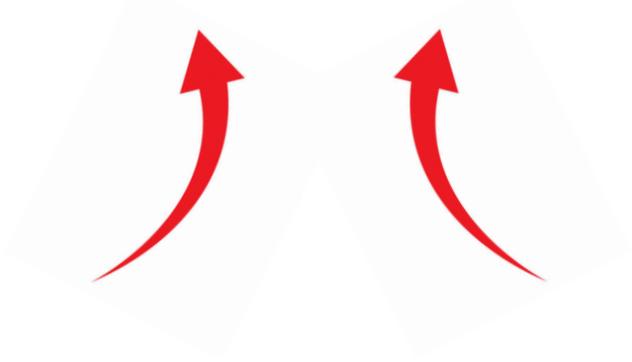
- 1) left + right
- 2)  $(1 + 2) + (3 + 4)$



# Bend: Fold



- 1) left + right
- 2)  $(1 + 2) + (3 + 4)$



# Bend: Установка



```
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

```
$ brew install gcc
```

```
$ cargo install hvm bend-lang
```

# Bend: Запуск



```
# Rust interpreter (sequential)
```

```
$ bend run main.bend -s
```

```
# C interpreter (parallel)
```

```
$ bend run-c main.bend -s
```

```
# CUDA (parallel)
```

```
$ bend run-cu main.bend -s
```

# Vend: Бенчи



- Язык сырой
- Поддержаны только 24-х битные числа
- Библиотек нет
- Бенчить можно только что-то \*простое

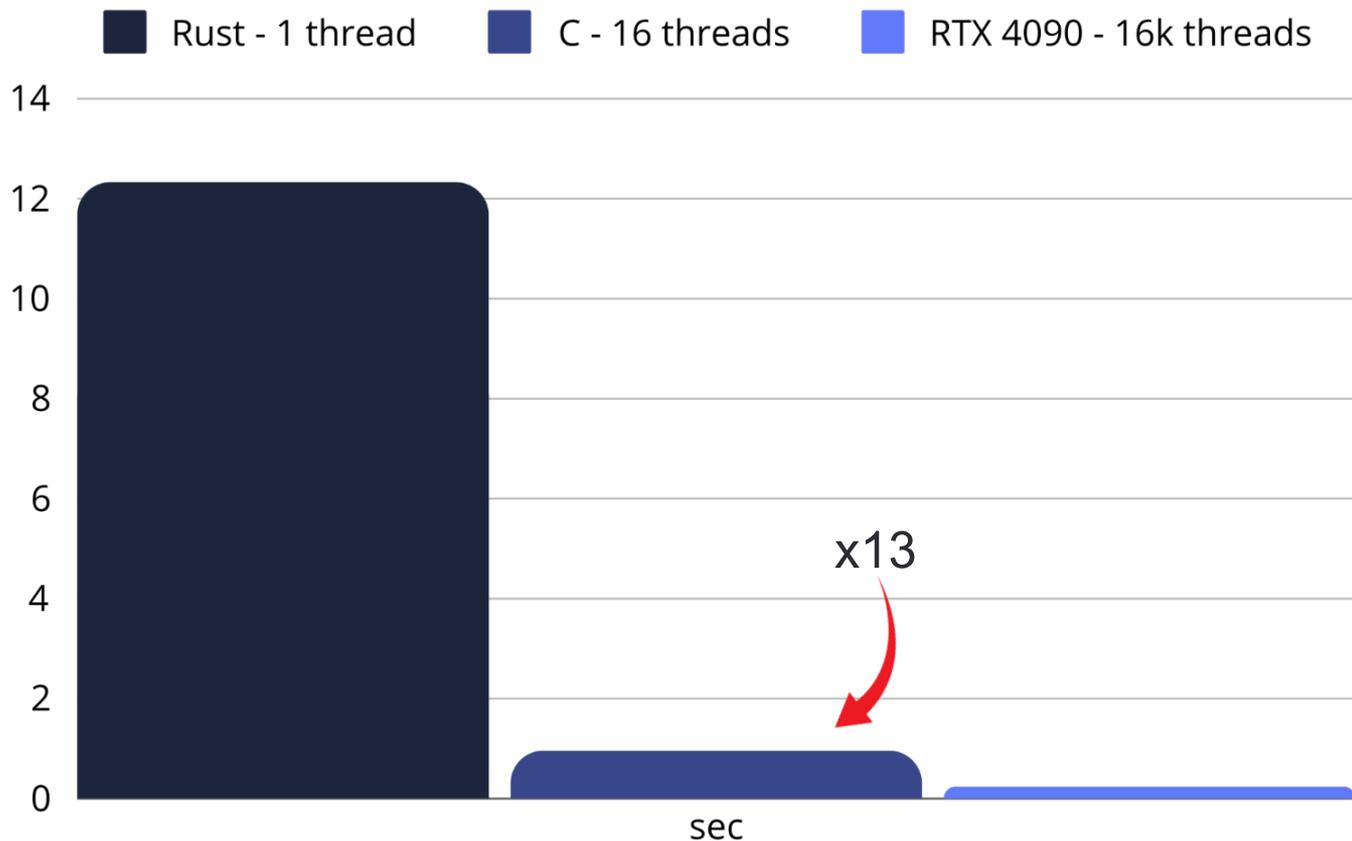
# Bend: Bitonic sort



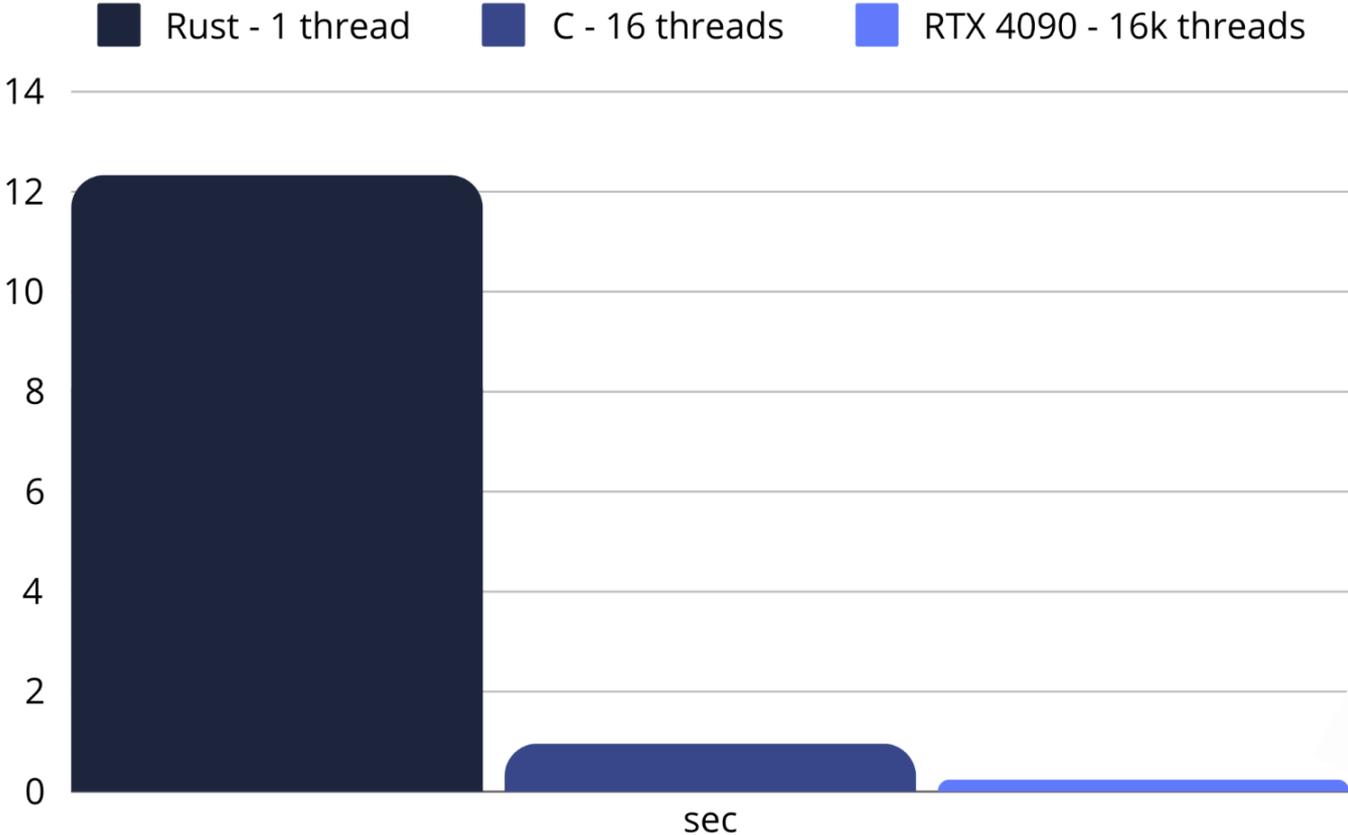
- Параллельный алгоритм
- Бьет массив на части
- Выполняется рекурсивно
- Подходит для GPU
- Apple M3 / RTX 4090



# Bend: Бенчи



# Bend: Бенчи



# Венд: Использование Python



- Пока что нельзя
- Но есть FFI
- Несовершенный компилятор
- Собрать будет сложно



# Bend: Эко-система



**Bend / GUIDE.md**

imaqt katt Update 'run' commands in guide 39bb8c6 · 2 weeks ago History

813 Lines (634 loc) · 24.3 KB

## Preview

### Bend in X minutes - the ultimate guide!

Bend is a high-level, massively parallel programming language. That means it feels like Python, but scales like CUDA. It runs on CPUs and GPUs, and you don't have to do anything to make it parallel: as long as your code isn't "helplessly sequential", it will use 1000's of threads!

While cool, Bend is far from perfect. In absolute terms it is still not so fast. Compared to SOTA compilers like GCC or GHC, our code gen is still embarrassingly bad, and there is a lot to improve. And, of course, in this beginning, there will be tons of instability and bugs. That said, it does what it promises: scaling horizontally with cores. And that's really cool! If you'd like to be an early adopter of this interesting tech, this guide will teach you how to apply Bend to build parallel programs in a new way!

For a more technical dive, check HVM2's [paper](#). For an entertaining, intuitive explanation, see HVM1's classic [HOW.md](#). But if you just want to dive straight into action - this guide is for you. Let's go!

### Installation

#### Install dependencies

##### On Linux

```
# Install Rust if you haven't it already.
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

# For the C version of Bend, use GCC. We recommend a version up to 12.x.
sudo apt install gcc
```

For the CUDA runtime [install the CUDA toolkit for Linux](#) version 12.x.

# Мојо: Сообщество



 @HigherOrderComp

 higherorderco

# Vend: Итоги



- Пока использовать нельзя
- Очень многообещающе
- Может улучшить питон
- Хочется еще побенчить



# Итоги



	Mojo	Bend
Концепция	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Опыт команды	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> / <input type="checkbox"/>
Скорость	<input checked="" type="checkbox"/>	!?
Удобство	<input type="checkbox"/>	<input type="checkbox"/>
Сообщество	<input type="checkbox"/>	×



 Mojo

Bend



 + Max

# Спасибо

