Ikra is an extended Ruby impl. w/data-parallel exec. (map & reduce over arrays) targeting GPGPU (& TSUBAME in future)

type inference supports integer, floating, bool, array, tuple, neighbor & proc. (plus user defined objects in future)

Goal of Ikra
quick prototyping of data-parallel programs integration with Ruby
- straightforward APIs
- minimal annotations
- compatibility with existing libraries

example: 2D diffusion + visualization

require 'sdl' # user requested library

# array initialization
a = Array.new_((SIZE,SIZE)) { |x,y| ...initialization... }
while true
  a = a.neighbor9.map( n | 
    n[ 0,0] + 2688
  )
  a = a.neighbor9.map( n | 
    n[ 0,0] + 2688
  )
  ...rest 7 neighbors);
  visualize

Type Inference

array(2,float) → int×→float
int×→float → array(2,float)
array(2,float) → int

Object → Object × Object → Object (unknown library calls are dynamic)

type inference vs. parallel skeletons

sequential code (Ruby)

visualize

Ruby + Ruby librs.
stock Ruby VM (CPU)

API

compatible w/Ruby's Array

PArray.new(50,50)

3D-diffusion performance relative to handwritten code

Performance

GUP

Optimizations (plan)

- separating boundary comp.
- spatial/temporal blocking
- loop fusion
- communication/computation overlapping
- designing a modular framework

~50% of handwritten CUDA code due to different implementation of boundary checking

~1000x faster than Ruby interpreter

~3D-diffusion performance relative to handwritten code

Handwritten

Ikra

Ikra

Goal of Ikra