Ikra: Leveraging Object-oriented Abstractions in a Ruby-to-CUDA JIT Translator

Matthias Springer, Hidehiko Masuhara
(Tokyo Institute of Technology)

Overview
- Acceleration of Ruby programs with GPUs (CUDA)
- High-level Goal: GPGPU for Ruby programmers
- Source code analysis and type inference at runtime

Restrictions in parallel sections:
- No restrictions
- No meta programming/reflection
- Dynamic typing should be avoided

Related Work:
- Job Reordering
  - Zheng et al.: On-the-fly elimination of dynamic irregularities for GPU computing. ASPLOS XVI.
- Kernel Fusion
  - Wahib et al.: Scalable kernel fusion for memory-bound GPU applications. SC '14.
  - Columnar Object Layout
  - Mathe et al.: Columnar objects improving the performance of analytical applications. ONWARD! 2015

Object Support
- Actor
  - @max_velocity: Float / rw
  - @progress: Float / rw
  - @street: Street / rw
- Pedestrian
  - move_ped: Int

Columnar Object Layout
Actor
- @max_velocity: Float / rw
- @progress: Float / rw
- @street: Street
- @color: Object

Street
- @length: Int

Heap Object Tracer
1. Type inference
   - Traverse only classes/methods reachable from main block
   - Determines if instance variables are read/written
   - Values type is not permitted and was not processed yet
2. Find objects and calc. column offsets
   - Object graph traversal. Follow instance variable if:
     - Instance variable is read/written
     - Variables are read/written
     - Traverse only classes/methods
3. Write object columns
   - Replace object references with offsets

Columnar Object Layout
- Idea: Represent all objects of a class as fields of arrays (columnar layout)
- Benefit: Chance for coalescing when accessing the same column in parallel
- Implementation: Heap Object Tracer converts object graph to columnar layout