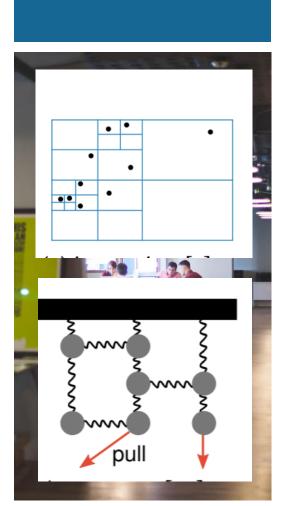
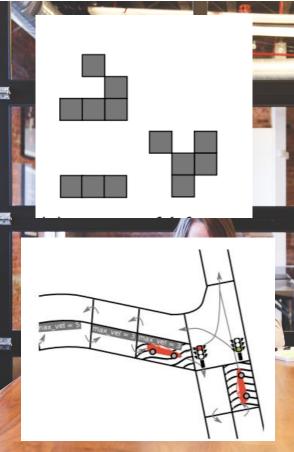
Hidehiko Masuhara Tokyo Tech (with Matthias Springer)

A Parallel
Object-Oriented
Programming
Language for
GPGPU









## **Concurrency model of GPGPU**



- At the low levels
  - SIMD (synchronized) execution
  - Distributed memory
- At the higher levels
  - No (or only coarse) synchronization
  - Globally shared memory
- State-of-the-art of programming style
  - Static data allocation + data-parallel

### Motivation: wanted a quick prototyping language for GPGPU



interested in GPGPU algorithms but CUDA/C is too naive...



 $\downarrow$ 

a scripting language for GPGPU!?









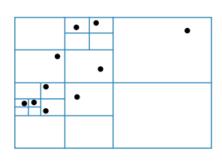
# Ikra: data-parallel extension to Ruby



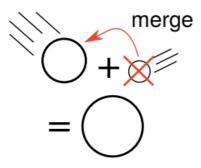
- Concurrency model: parallel arrays with map & reduce
- OOP support
  - Optimized memory layout for GPU
  - Parallel arrays (or sets) of objects
  - Dynamic object allocation ←new!
     [ECOOP'19, ISMM'19]

### Parallel OOP Applications (demo)

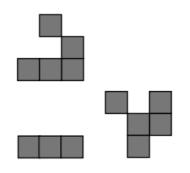




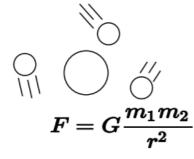
(a) barnes-hut [4]: Parallel Tree Constr.



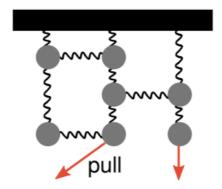
**(b)** collisions: Particle System



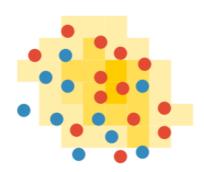
**(c)** game-of-life: Cellular Automaton



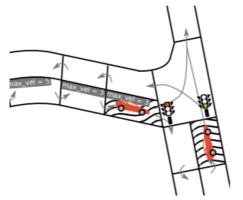
**(d)** nbody: Particle System



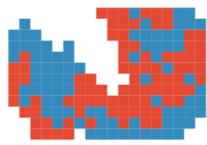
**(e)** structure [14]: Finite Elem. Method



**(f)** sugarscape [8]: Agent-based Sim.



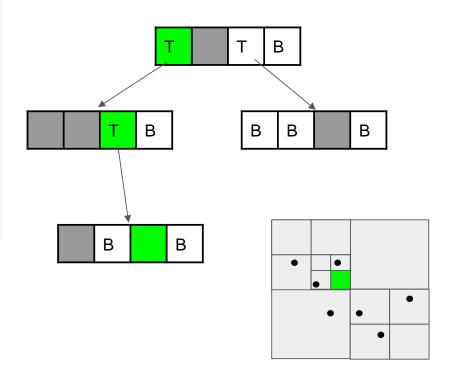
**(g)** traffic [17]: Nagel-Schr. Model

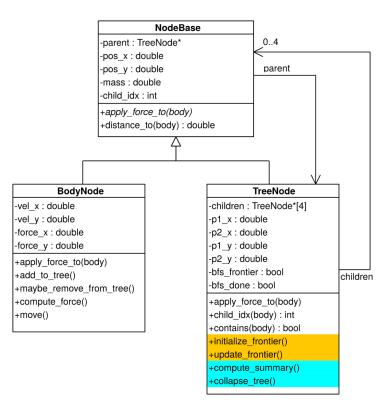


**(h)** wator [6]: Agent-based Sim.

# Dynamic allocator enables parallel tree construction

#### Example 3: Data Structure





# OOP helps programming complicated applications



### Eg. Traffic simulation

- Road Segment
  - Regular / Junction Expressway / ...
- Vehicle
  - Regular / Bus / Emergency / ...



### **Dynamic memory allocation inside of GPU**

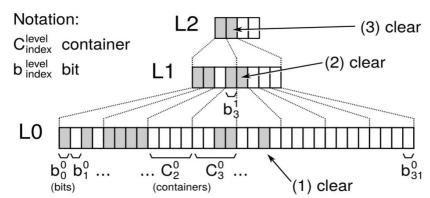


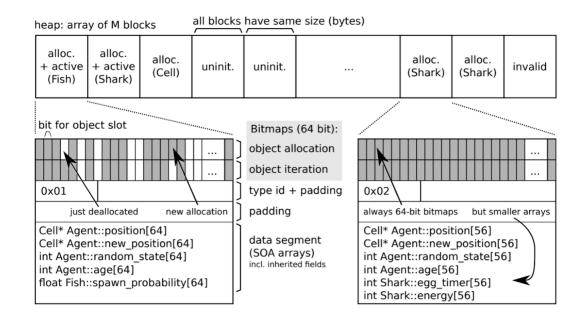
#### Challenges

- No global locks
- Keep allocated data dense

#### **Approaches**

- Block based
- Hierarchical bitmaps





### Summary



alloc.

always 64-bit bitmans but smaller arrays

Cell\* Agent::position[56]

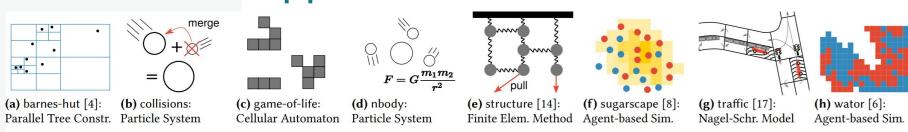
int Shark::egg\_timer[56]
int Shark::energy[56]

int Agent::age[56]

Cell\* Agent::new position[56]

int Agent::random\_state[56]

 OOP helps programming complicated GPGPU applications



A lot of "standard" mechanisms need to be reworked for GPGPU
heap: array of M blocks have same size (bytes)

alloc

- active

(Shark)

just deallocated

Cell\* Agent::position[64]

int Agent::age[64]

Cell\* Agent::new\_position[64]

float Fish::spawn\_probability[64]

int Agent::random\_state[64]

(Fish)

0x01

bit for object slot

uninit.

uninit

Bitmaps (64 bit): object allocation

object iteration type id + padding

data segment

(SOA arrays)

padding

0x02

