東京工業大学
Tokyo Institute of Technology

CompactGpu: Massively Parallel Memory Defragmentation on GPUs

Matthias Springer

Tokyo Institute of Technology

ACM SRC @ PLDI 2019

Introduction / Motivation

東京工業大学
Tokyo Institute of Technology

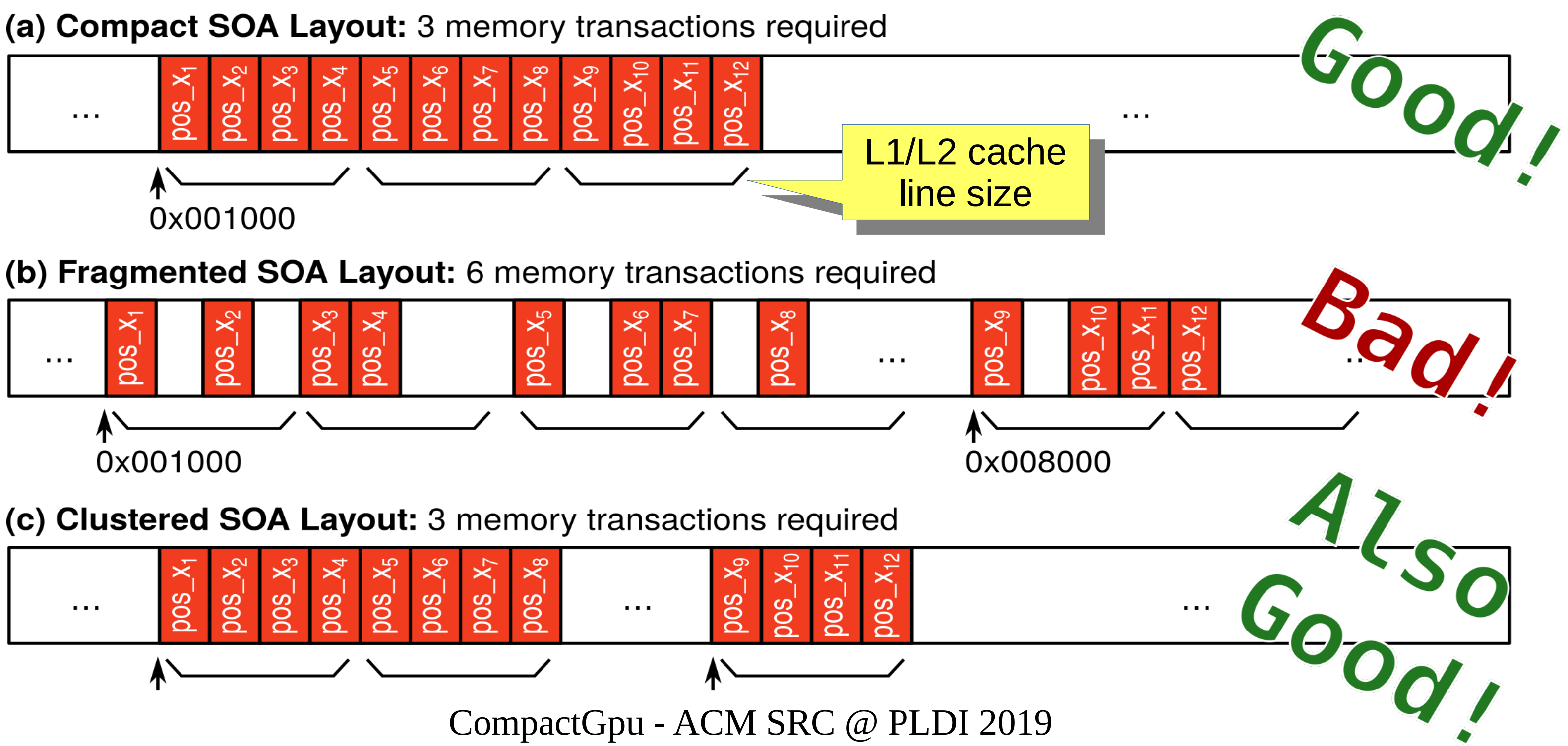
- *Goal:* Make GPU programming easier to use.
- *Focus:* Object-oriented programming on GPUs/CUDA.
  - Many OOP applications in high-performance computing.
  - DynaSOAr [1]: Dynamic memory allocator for GPUs.
  - **CompactGpu:** Make allocations more space/runtime efficient with memory defragmentation.

[1] M. Springer, H. Masuhara. **DynaSOAr: A Parallel Memory Allocator for Object-oriented Programming on GPUs with Efficient Memory Access.** ECOOP 2019.

Why Defragment GPU Memory?

東京工業大学
Tokyo Institute of Technology

- *Space Efficiency:* Reduce overall memory consumption (and prevent premature out of memory errors).
- *Runtime Efficiency:* Accessing compact data requires **fewer vector transactions** and benefits cache utilization.



東京工業大学
Tokyo Institute of Technology

GPU Allocation Characteristics

- **Massive number** of concurrent allocations.
- Most allocations are small and have the **same size** (due to mostly uniform control flow).
- Allows us to optimize defragmentation more than on CPUs.

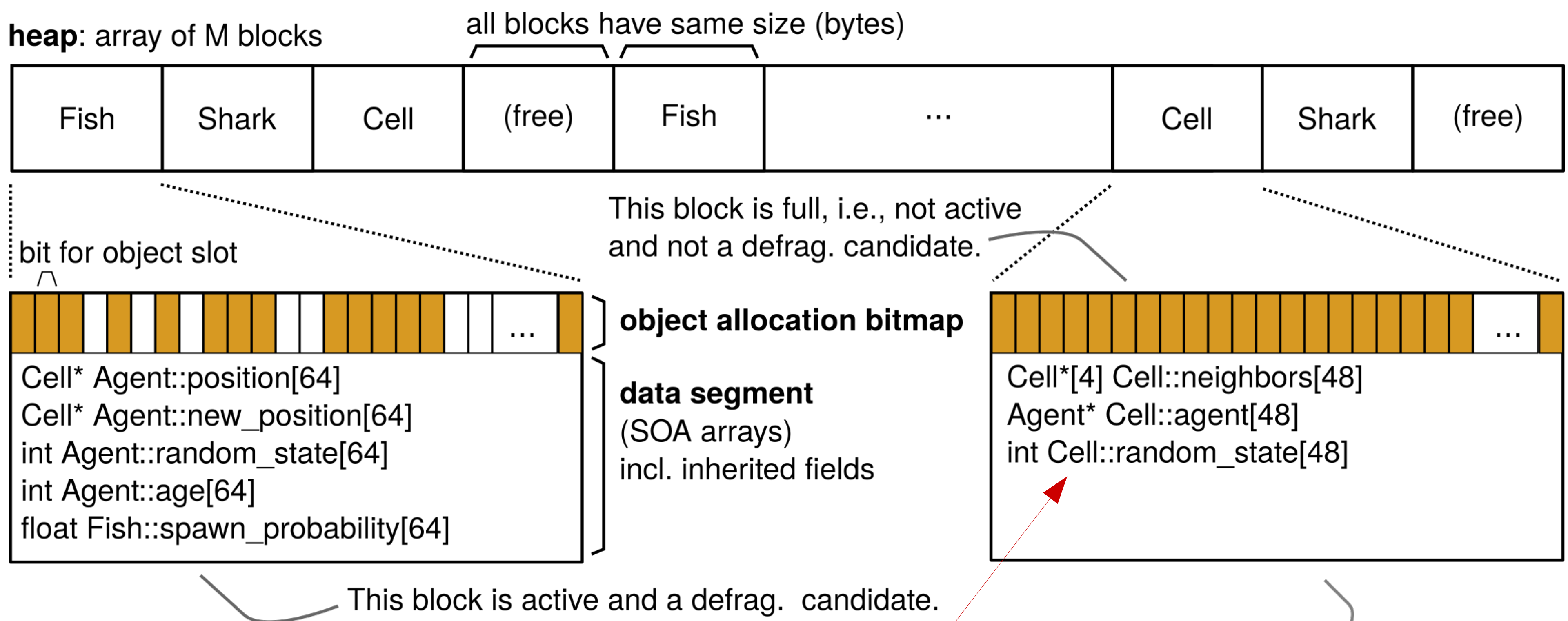
Related Work / State of the Art

東京工業大学
Tokyo Institute of Technology

- Dynamic GPU Memory Allocation
  - Not well supported until recently, so not widely utilized yet.
  - Default CUDA allocator (malloc/free): Unoptimized and extremely slow.
  - Halloc [2], ScatterAlloc/mallocMC [3]: Fast (de)allocation time, but high fragmentation (hashing).
  - DynaSOAr: My own allocator, with additional optimizations for structured data (objects).
- GPU Memory Defragmentation [4]
  - High runtime overhead (up to 50%).
  - Different assumptions about allocation pattern.
  - Uses a memory allocator for moving allocations in memory.

[4] R. Veldema, M. Phillipsen. **Parallel Memory Defragmentation on a GPU.** MSPC 2012.

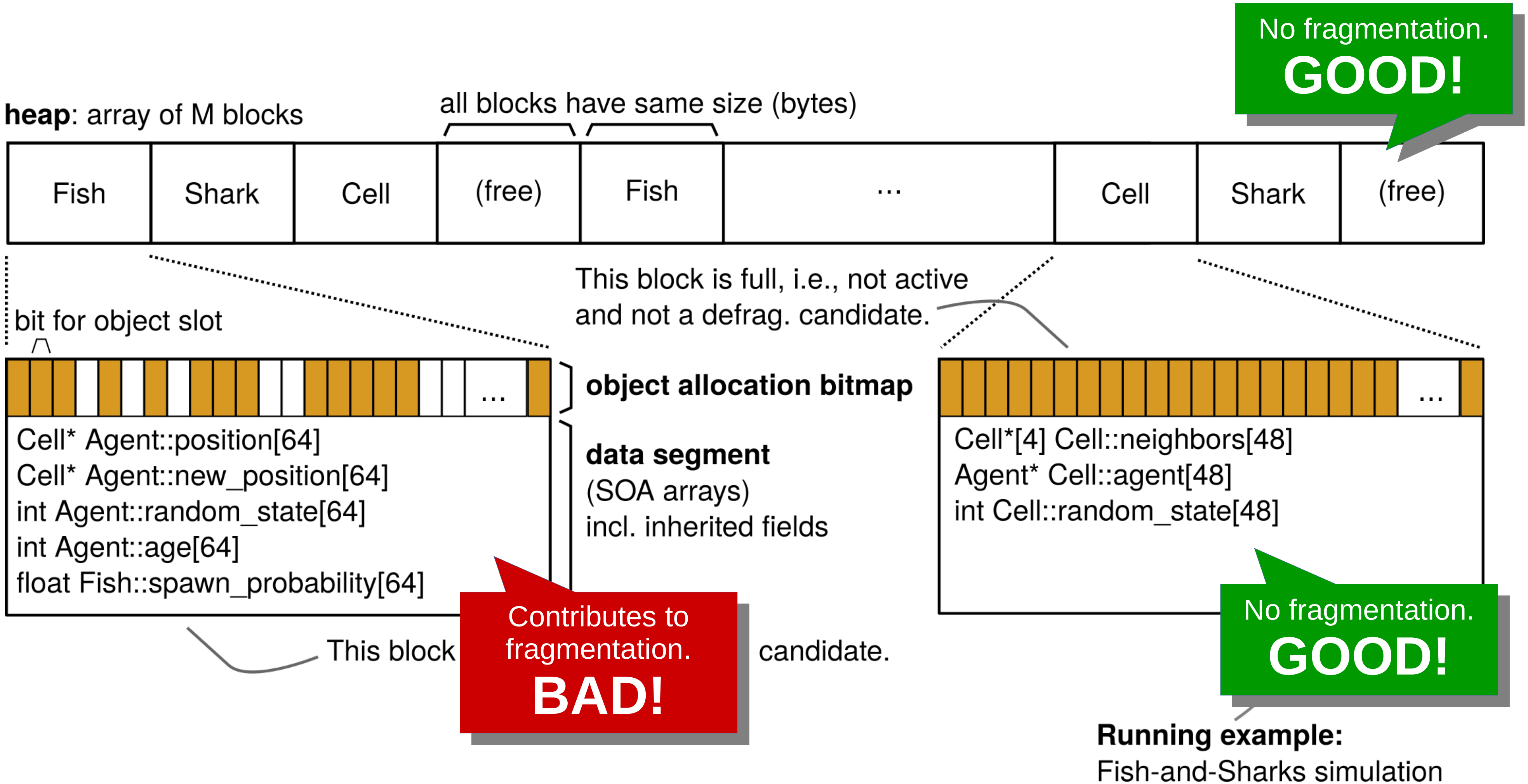
DynaSOAr Heap Layout



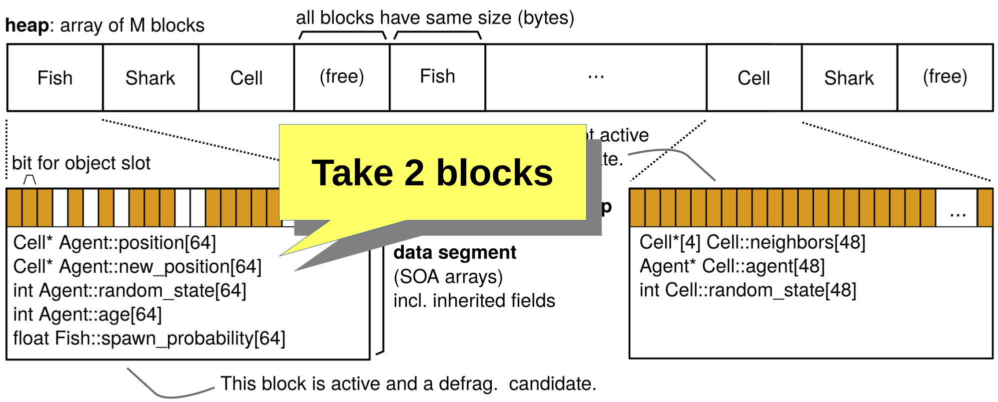
Contains 48 objects in **Structure of Arrays** (SOA) data layout

Running example:
Fish-and-Sharks simulation

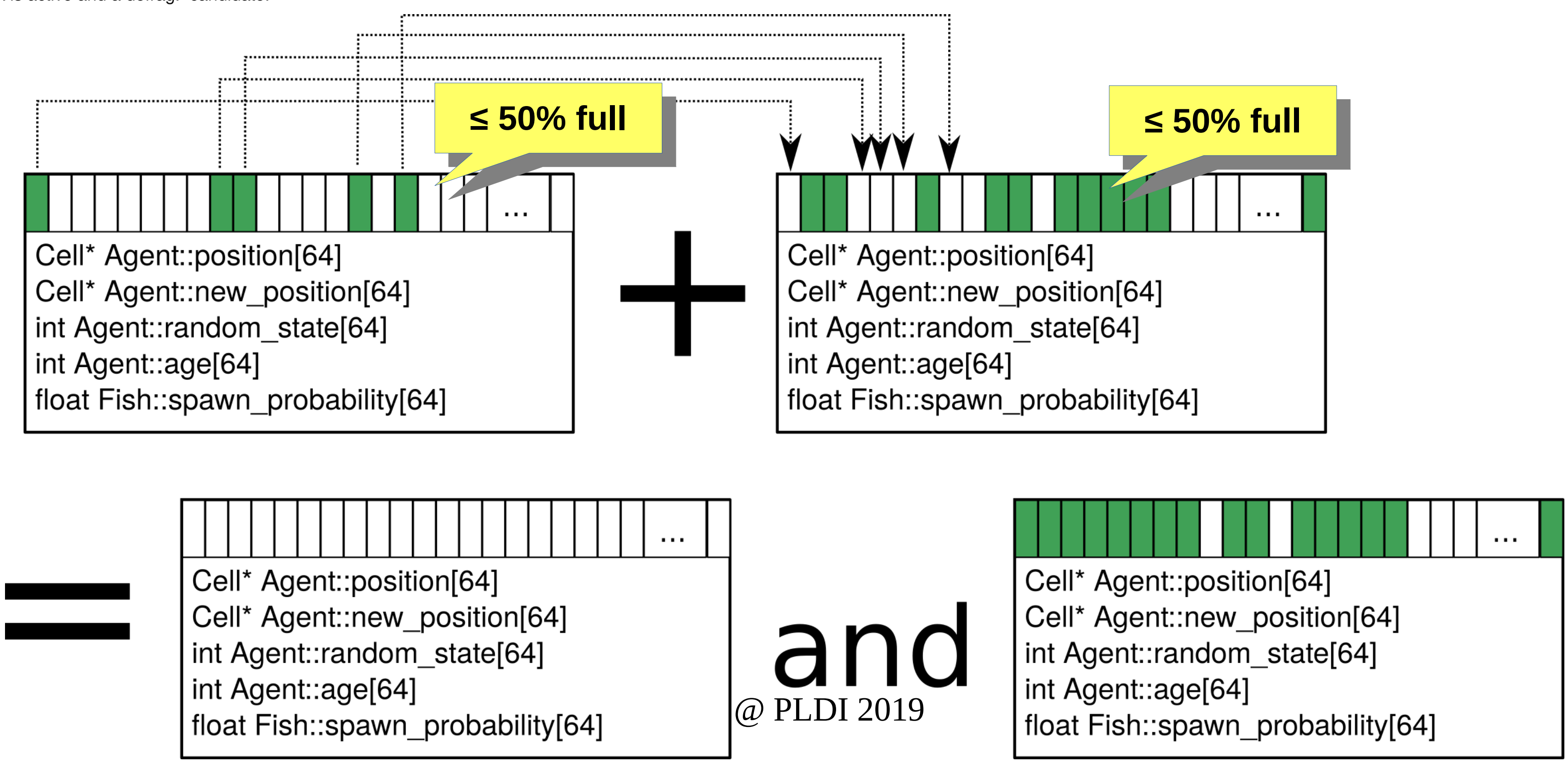
DynaSOAr Heap Layout



Block Merging: 1 + 1 = 1

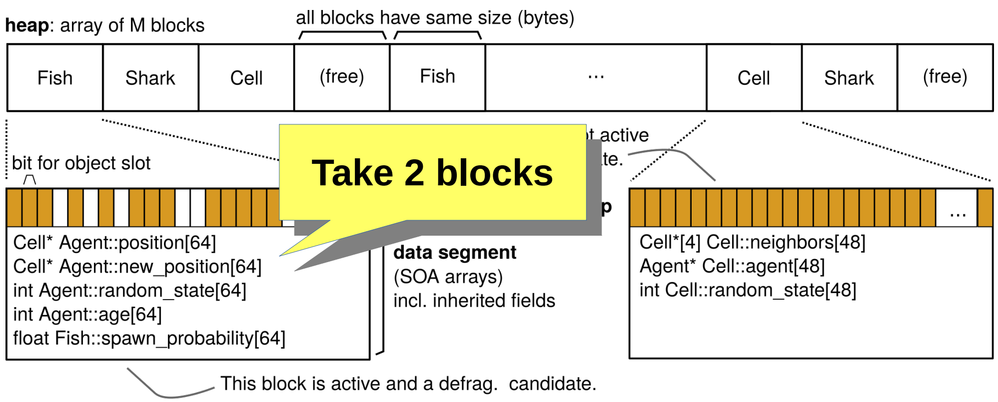


Do this
in parallel
for all
eligible
blocks:

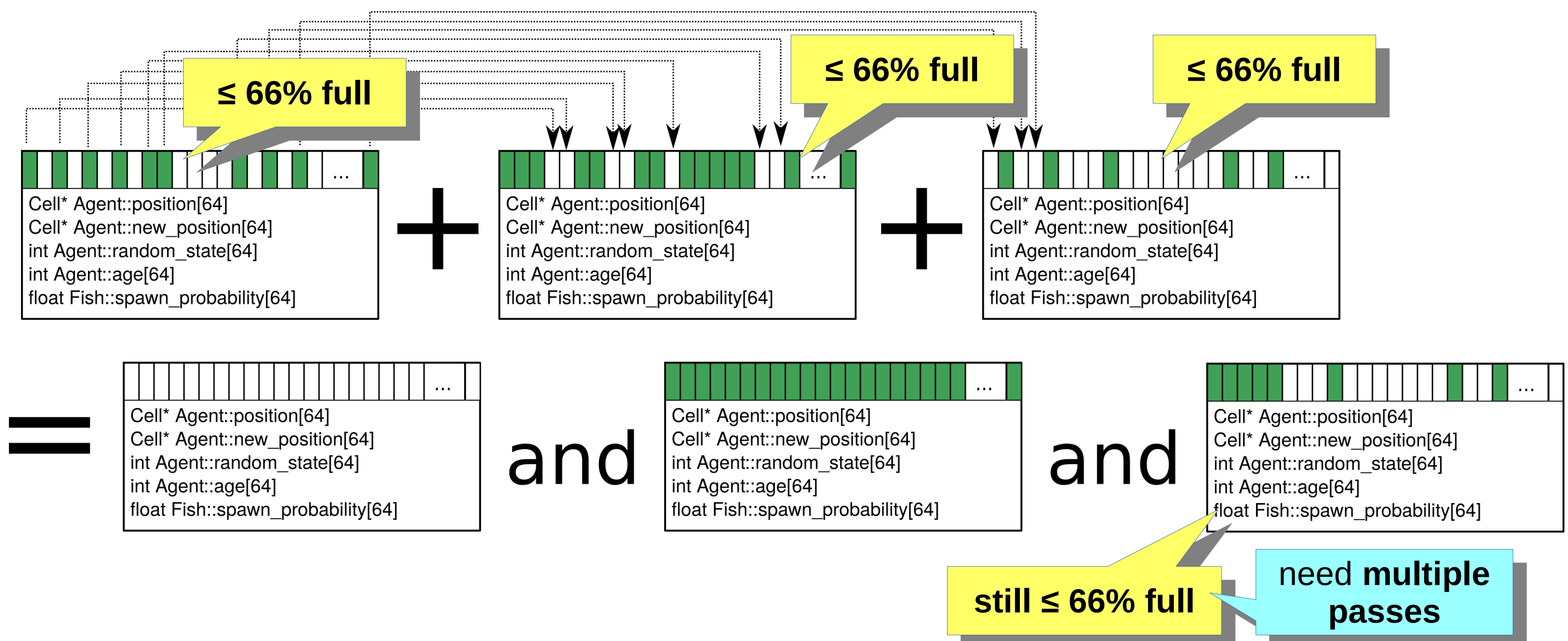


東京工業大学
Tokyo Institute of Technology

Block Merging: 1 + 2 = 2

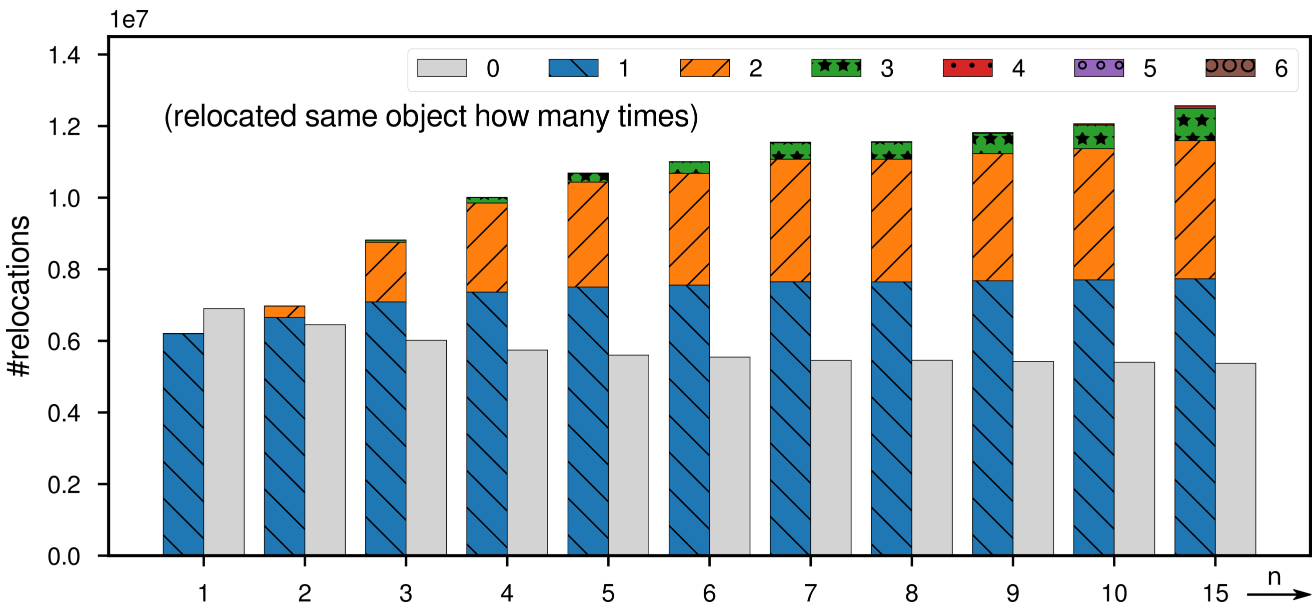


Do this
in parallel
for all
eligible
blocks:



Block Merging: 1 + n = n

- Higher *n*: Better defragmentation guarantees.
- Lower *n*: A bit faster, fewer passes.
- *n* is can be configured by the programmer.



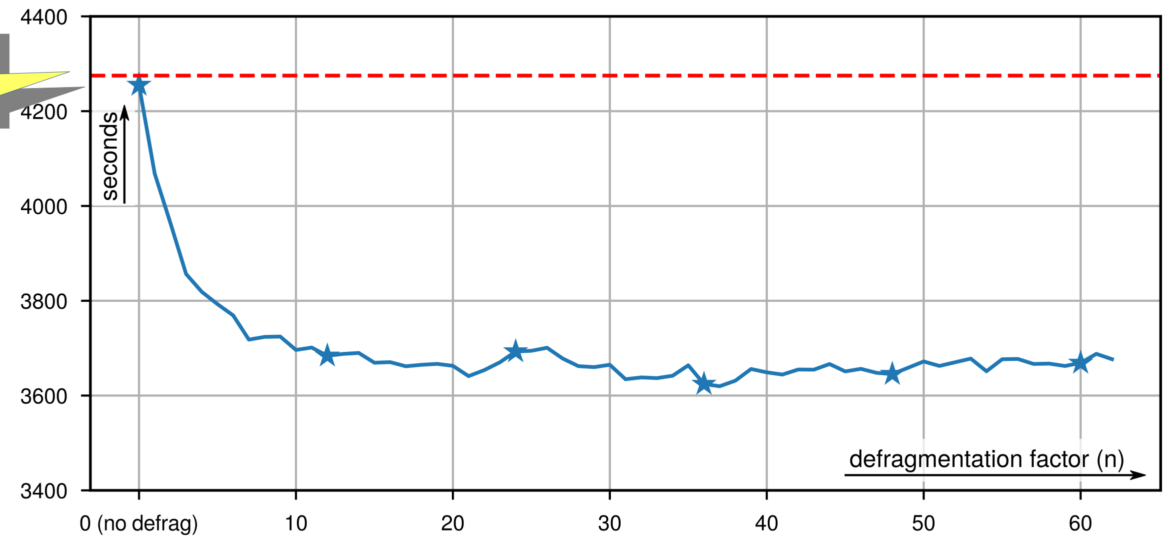
Pointer Rewriting

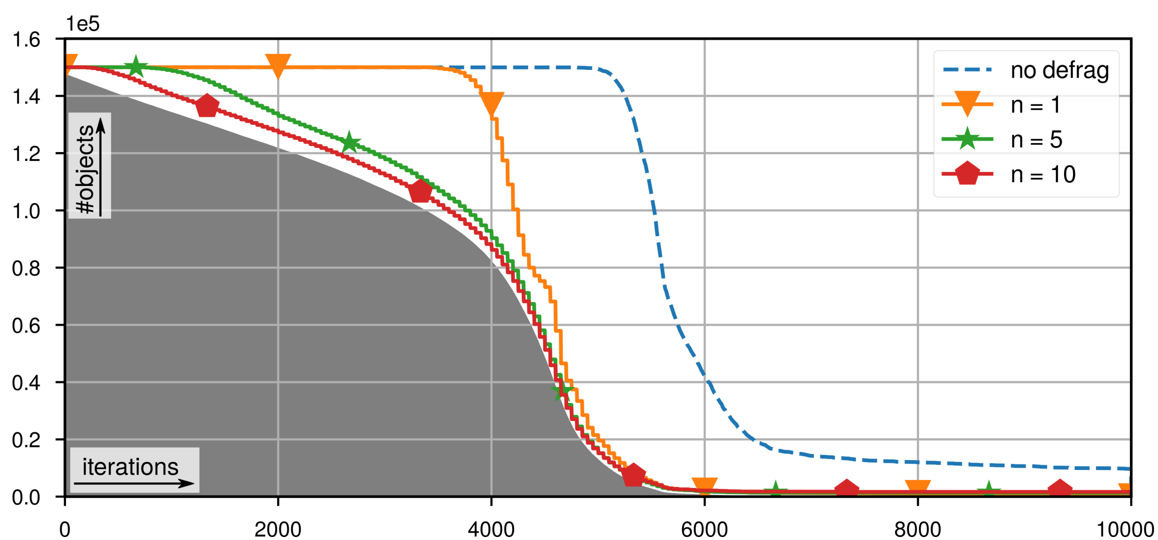
東京工業大学
Tokyo Institute of Technology

- Rewrite pointers to objects that were moved.
- Basic Ideas:
  - Store **forwarding pointers** in source blocks.
  - Allocator has knowledge about the structure (fields, classes) of the data it is allocating. **No need to scan the entire heap**.
  - Quickly decide if a pointer must be rewritten with **bitmaps** that fit in the L2 cache.

Benchmark Results: n-body

red line:
no defrag





This benchmark:
Defragmentation is about **0.5%** of the the total running time.

Conclusion

- Efficient memory defragmentation is **feasible on GPUs**.
- Besides saving memory, defragmentation makes usage of allocated memory more efficient: **Better cache utilization and better vectorized access**.
- GPU allocation patterns allow us to implement defragmentation very efficiently.
  - *Choosing source/target blocks:* Parallel prefix sum.
  - *Copying objects:* Very efficient due to SOA layout.
  - *Rewriting pointers:* Fast due to many optimizations that reduce #memory accesses (bitmaps, restricting heap scan areas).

東京工業大学
Tokyo Institute of Technology

References

[1] M. Springer, H. Masuhara. **DynaSOAr: A Parallel Memory Allocator for Object-oriented Programming on GPUs with Efficient Memory Access.** ECOOP 2019.

[2] A. V. Adinetz and D. Pleiter. **Halloc: A High-Throughput Dynamic Memory Allocator for GPGPU Architectures.** GPU Technology Conference 2014.

[3] M. Steinberger, M. Kenzel, B. Kainz, D. Schmalstieg. **ScatterAlloc: Massively Parallel Dynamic Memory Allocation for the GPU.** InPar 2012.

[4] R. Veldema, M. Phillipsen. **Parallel Memory Defragmentation on a GPU.** MSPC 2012.