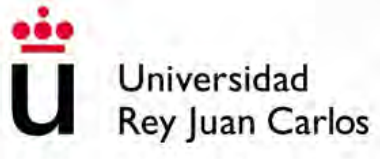
The Impact of Memory Management on OpenFoam Performance

David Concha, José Ángel Moreno San Segundo,
Gabriel M. Magalhães, J. Miguel Nóbrega



AGENDA

1. Motivation
2. Open Foam Memory Usage
3. Memory Pool Design
4. Experimental Results
5. Conclusions and Future Work

AGENDA

1. **Motivation**
2. Open Foam Memory Usage
3. Memory Pool Design
4. Experimental Results
5. Conclusions and Future Work

Motivation

- OpenFoam: **Field**, Operation And Manipulation
- Fields memory requirements are centralized in the **List class**.
- Modifying List we can change the **memory management**.

AGENDA

1. Motivation
2. **Open Foam Memory Usage**
3. Memory Pool Design
4. Experimental Results
5. Conclusions and Future Work

OpenFoam Memory Usage

- In the normal use of OpenFoam **many allocations** are made
  - Allocations for **temporal** results are often overlooked

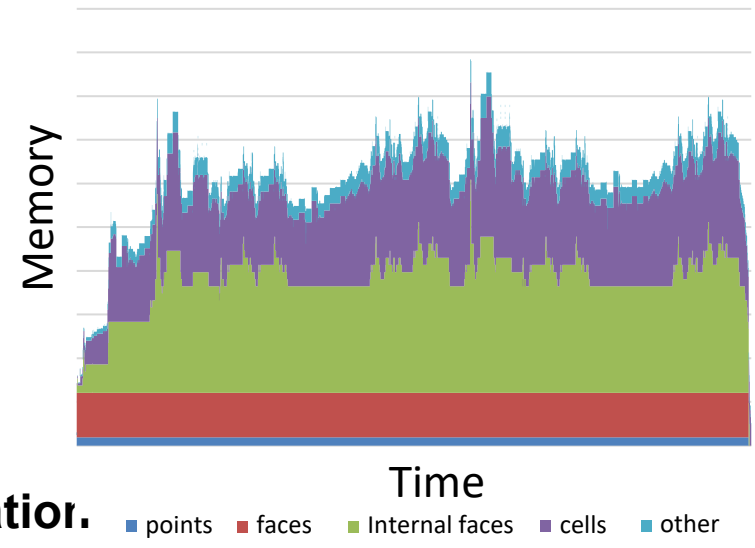
```
r = A + B * C;

r = A + <temp>;

r = <temp>;
```

- This happens with many types of data such as **scalars**, **vectors**, **tensors**, etc.

OpenFoam Memory Usage

- In the chart:
  - Every rise is an allocation
  - Every fall is a free
  - Two iterations are shown

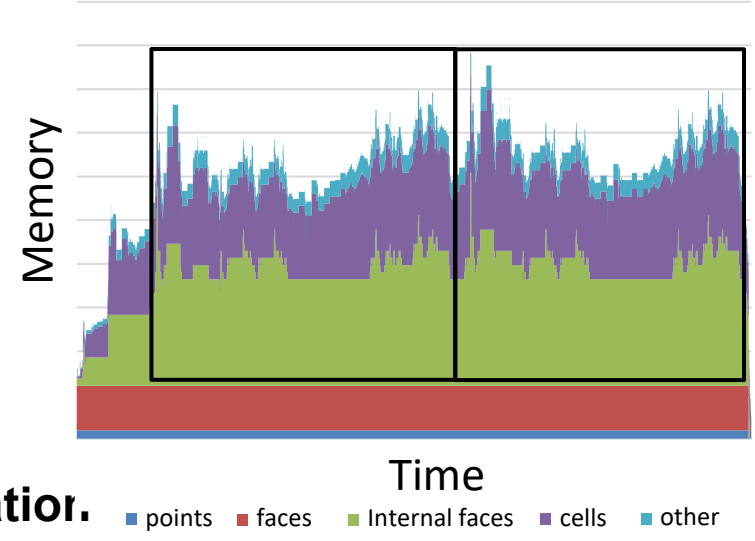
- **Our plan**:
  - Do the allocations on the **first iteration**.
  - **Never free** memory
  - **Reuse** already allocated memory



OpenFoam Memory Usage

- In the chart:
  - Every rise is an allocation
  - Every fall is a free
  - Two iterations are shown

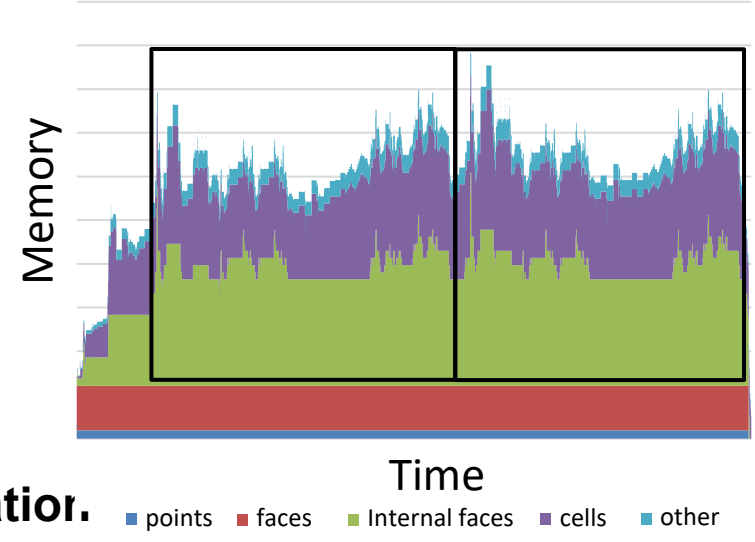
- **Our plan**:
  - Do the allocations on the **first iteration**.
  - **Never free** memory
  - **Reuse** already allocated memory



OpenFoam Memory Usage

- In the chart:
  - Every rise is an allocation
  - Every fall is a free
  - Two iterations are shown

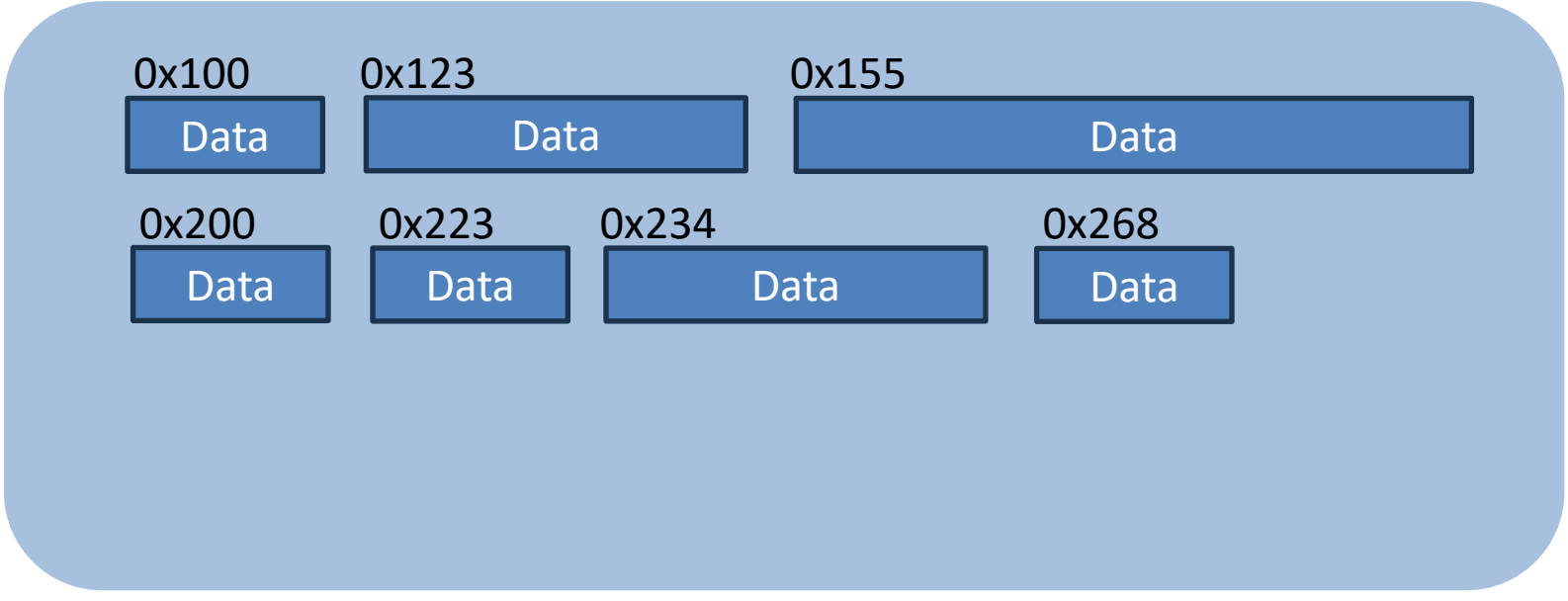
- **Our plan**:
  - Do the allocations on the **first iteration**.
  - **Never free** memory
  - **Reuse** already allocated memory



~27k allocations and frees per iteration

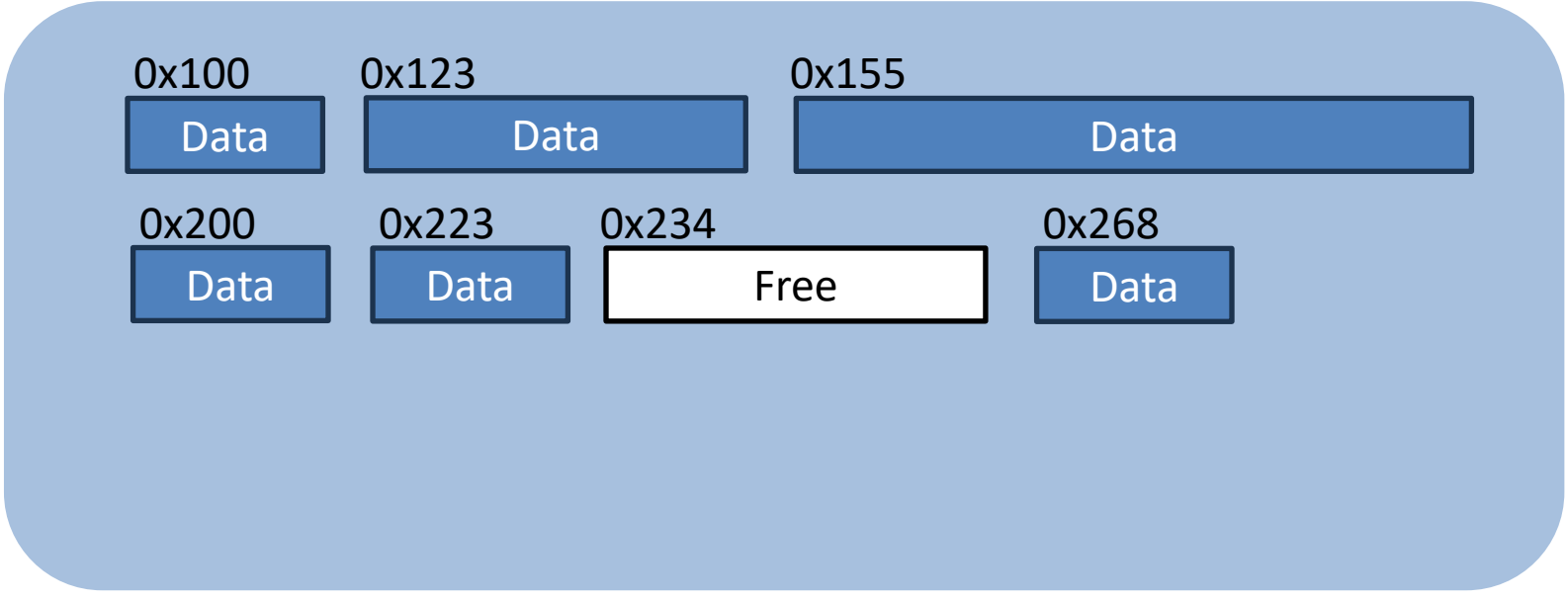
OpenFoam Memory Usage

- OpenFoam memory footprint



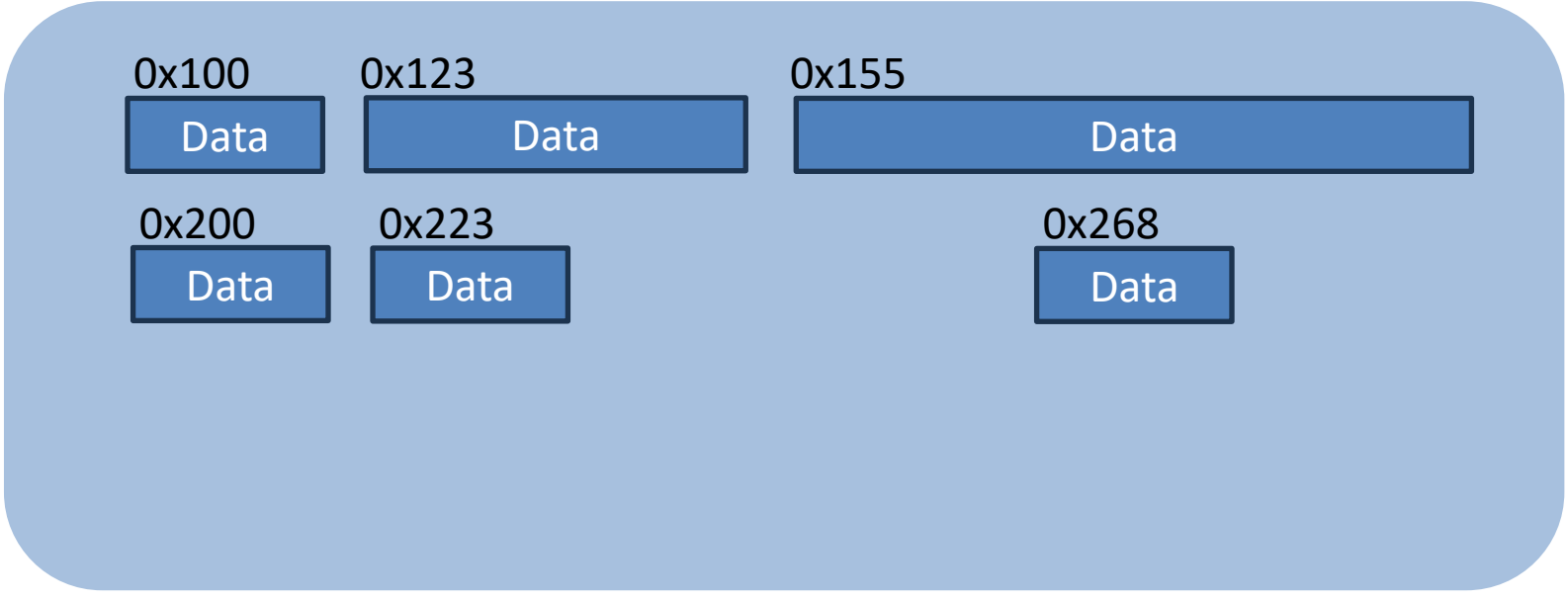
OpenFoam Memory Usage

- OpenFoam memory footprint



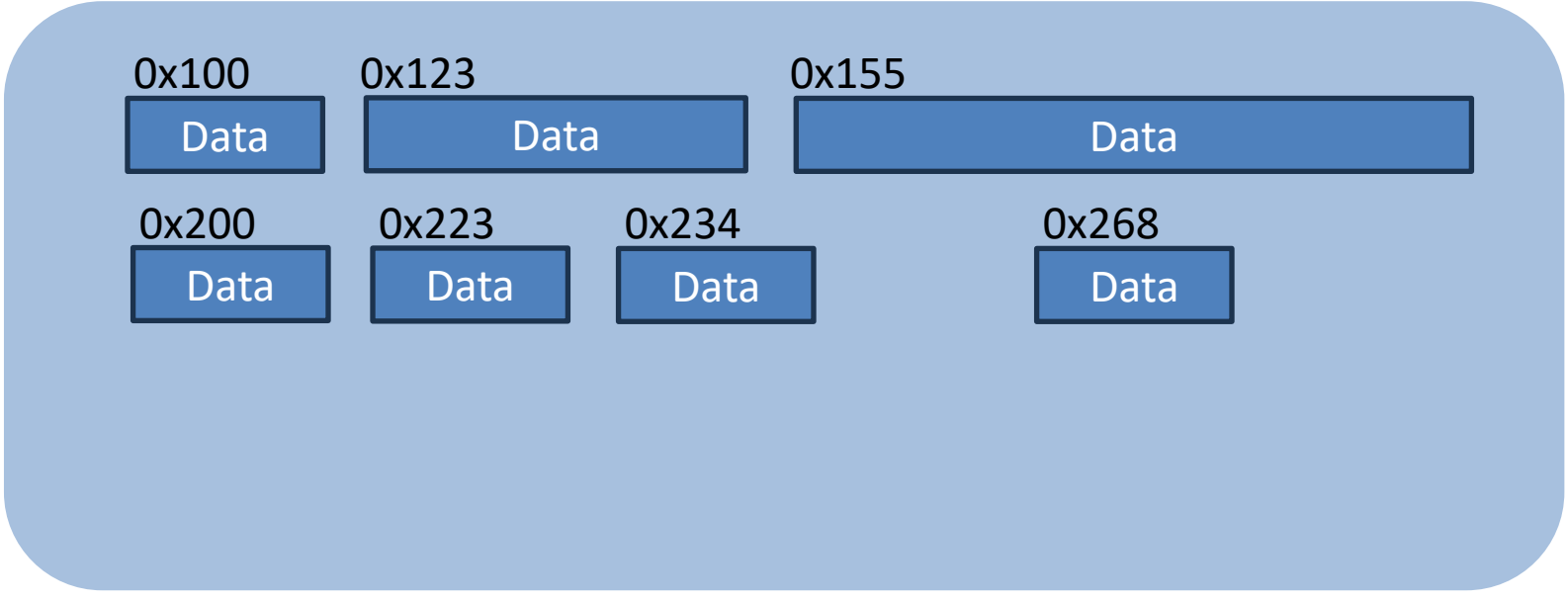
OpenFoam Memory Usage

- OpenFoam memory footprint



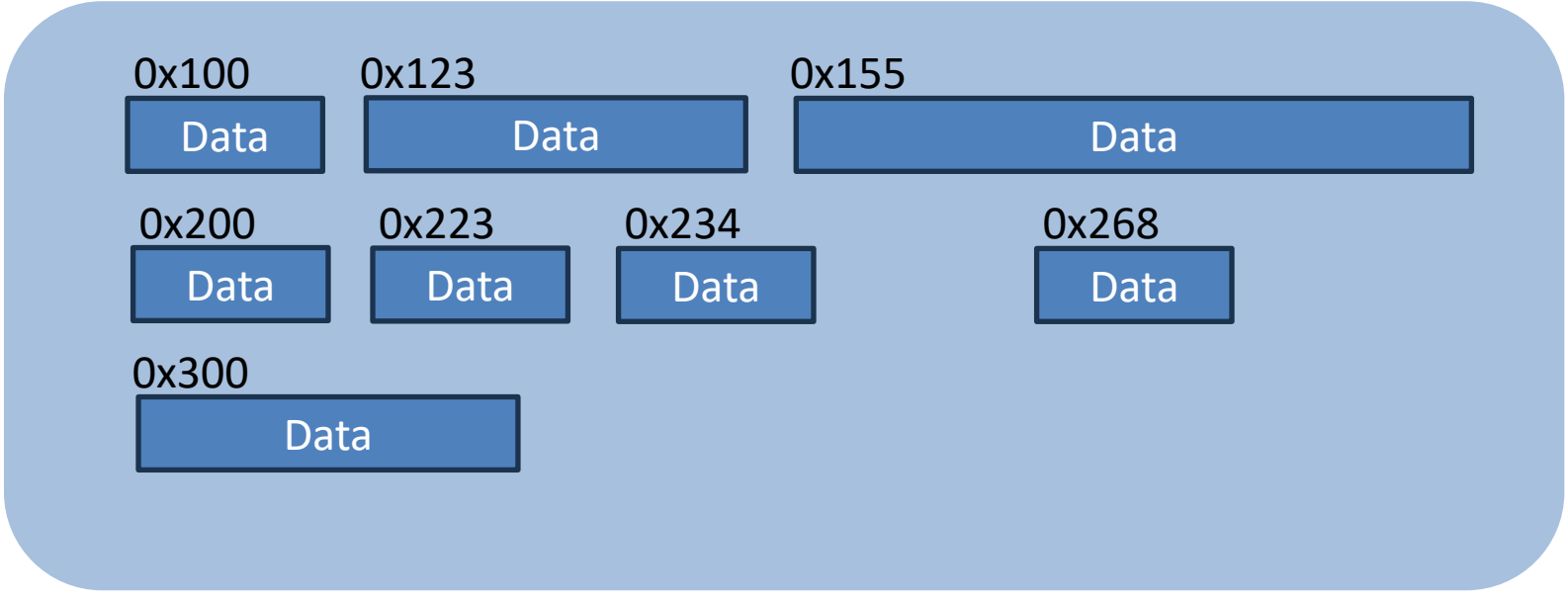
OpenFoam Memory Usage

- OpenFoam memory footprint



OpenFoam Memory Usage

- OpenFoam memory footprint



AGENDA

1. Motivation
2. Open Foam Memory Usage
3. **Memory Pool Design**
4. Experimental Results
5. Conclusions and Future Work

Memory Pool Design

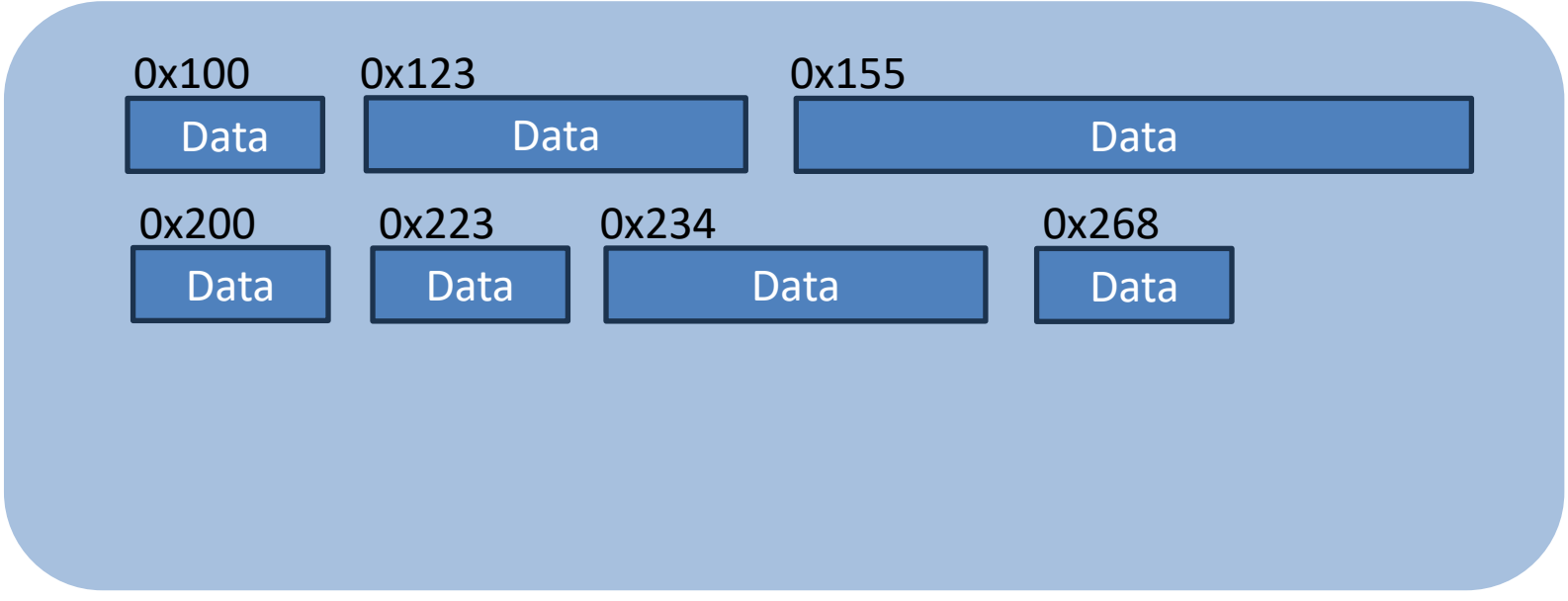
- The general idea is to make de **allocations, reuse and never free**.
  - Keep track of **“unused”** memory blocks.

- Foam::List:
  - Allocations mainly in `alloc`, `realloc` and `setSize`
  - Frees in `clear` and `destructors`.

- Memory pool:
  - If there is an “unused” memory block, it **reuses** it, otherwise it makes an **allocation**.
  - On frees, marks the memory block as **“unused”** but does **not free** the memory.

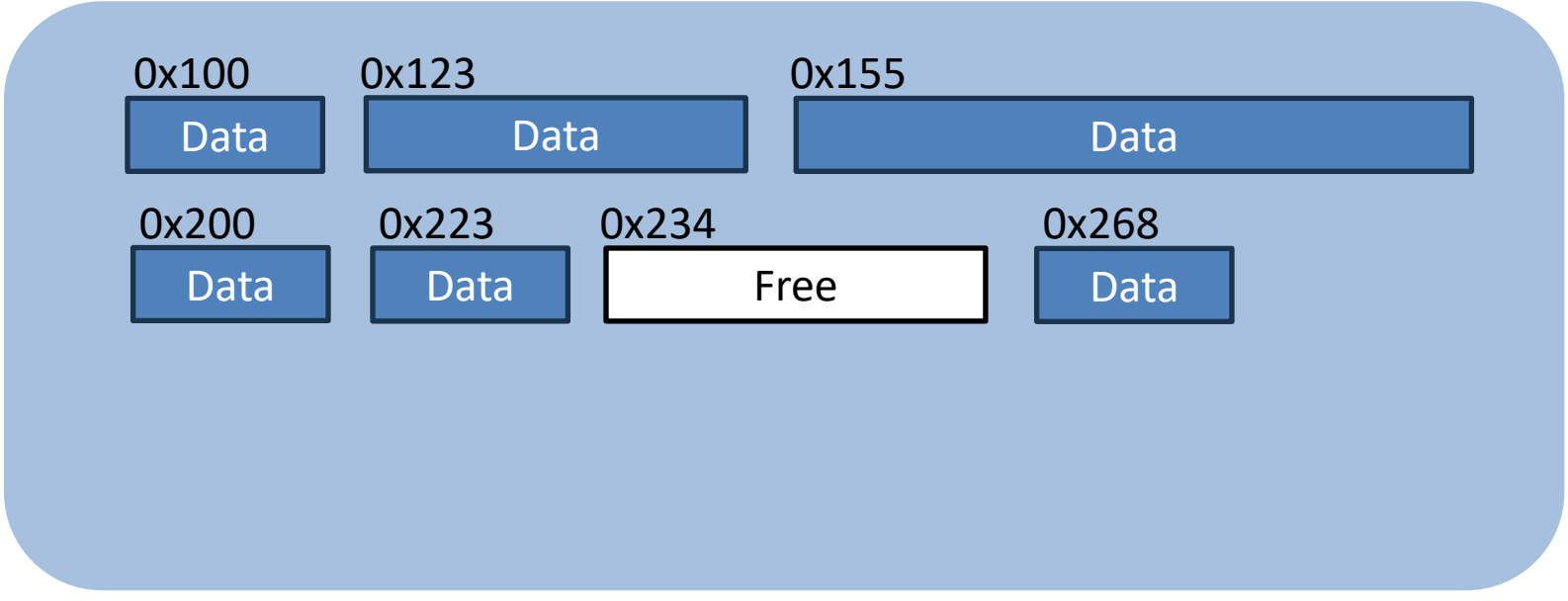
Memory Pool Design

- Our memory footprint



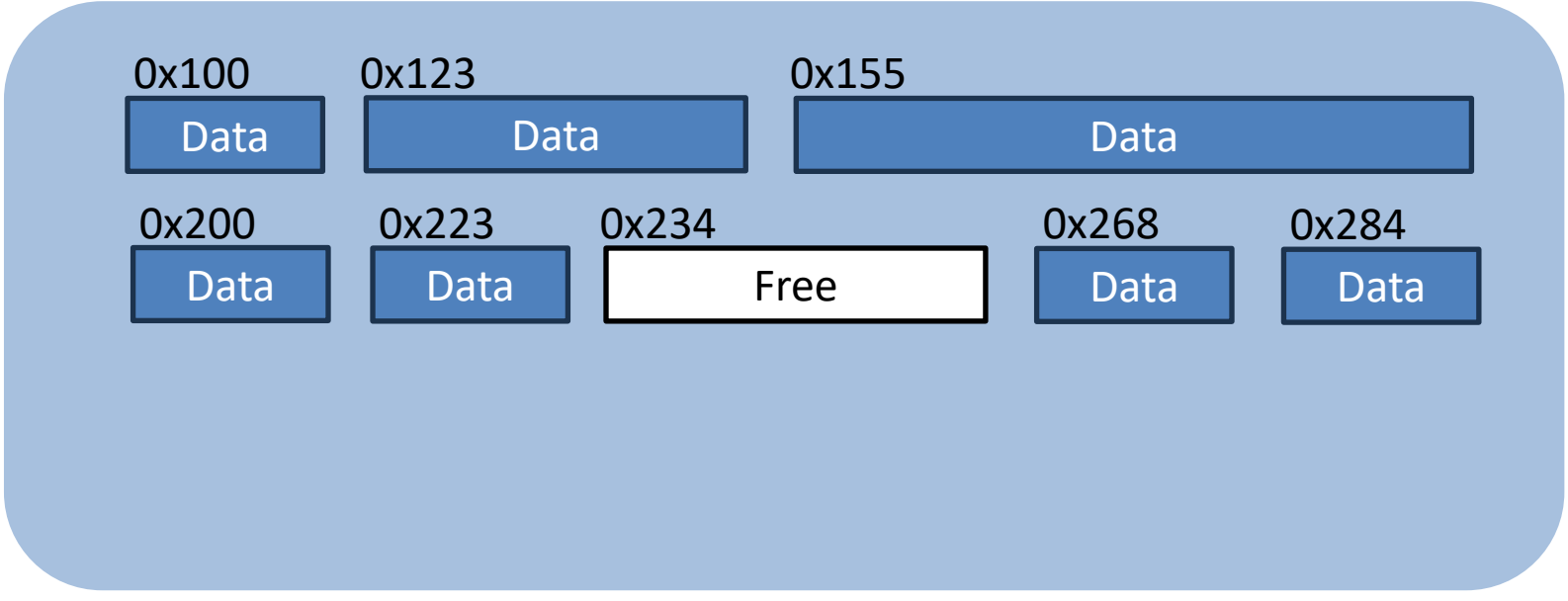
Memory Pool Design

- Our memory footprint



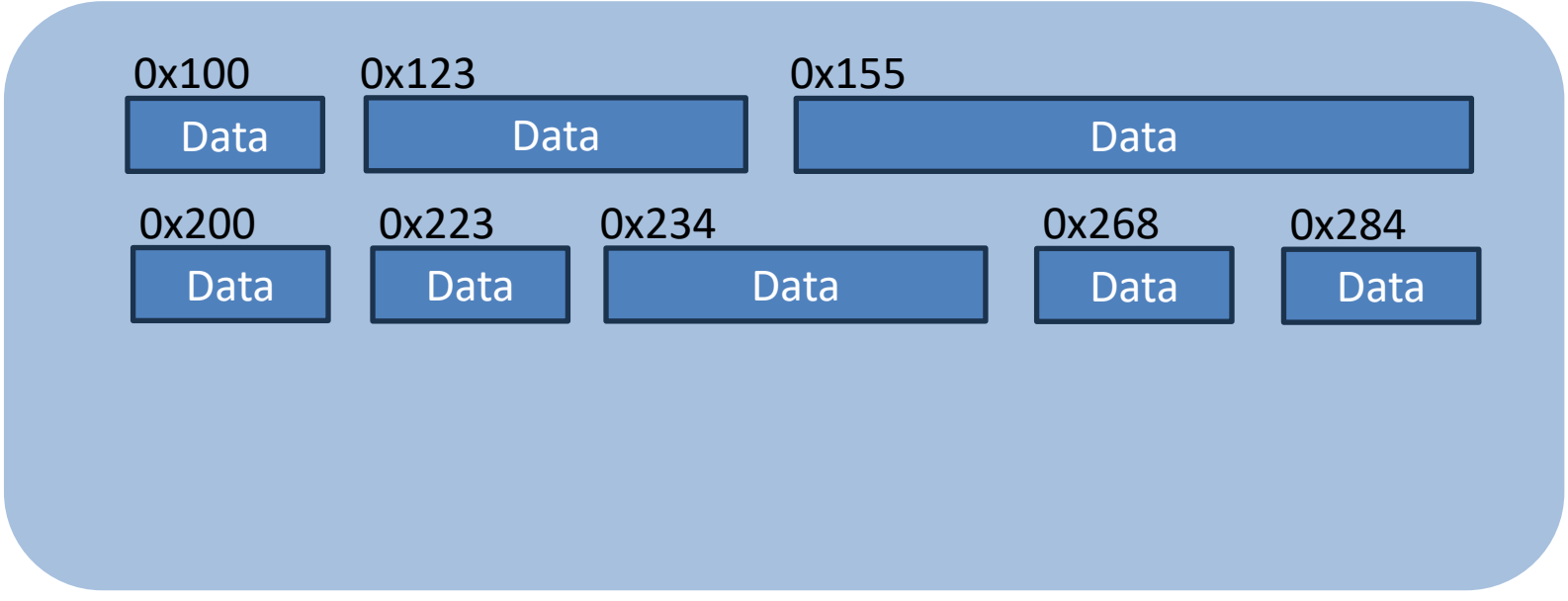
Memory Pool Design

- Our memory footprint



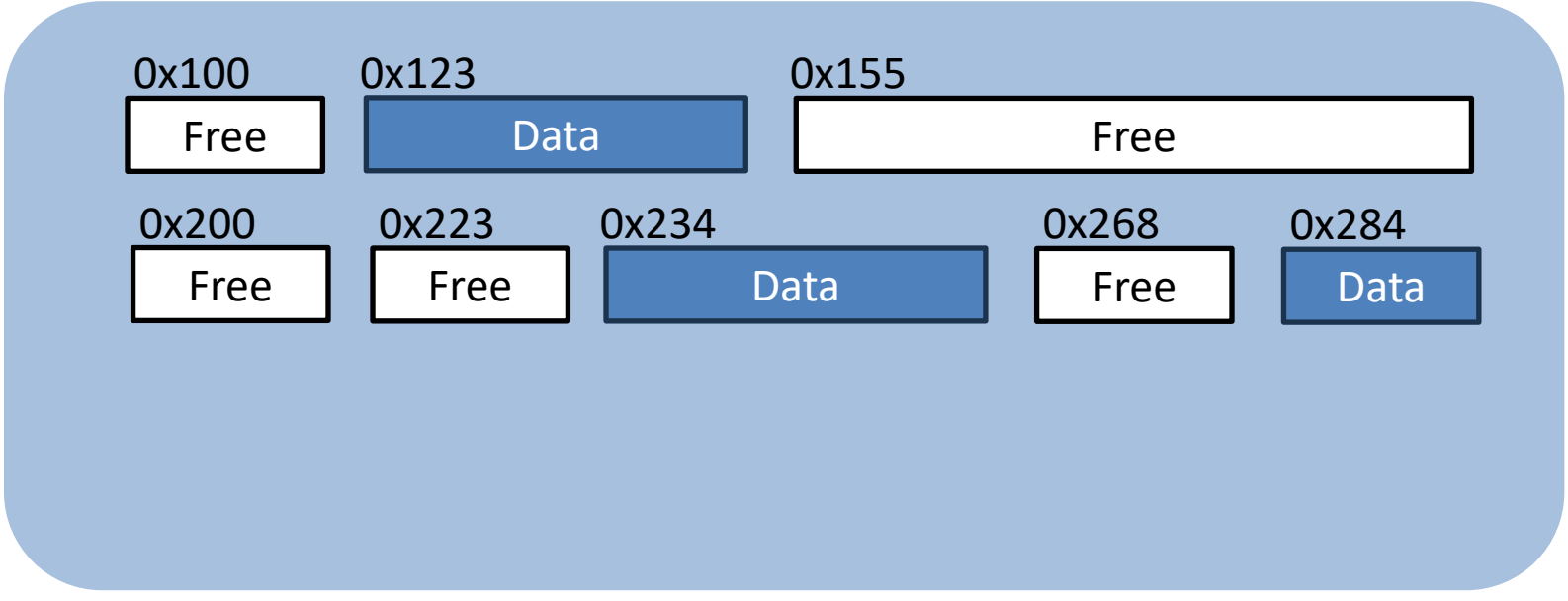
Memory Pool Design

- Our memory footprint



Memory Pool Design

- Our memory footprint



Memory Pool Design

- We thought we need store the pointers to the allocated memory blocks.
  - `std::deque` over `std::vector`
  - Use `std::unique_ptr` for automatic free when the program finish.
- Use a linked list to **store "unused" memory blocks** (pointers)
- We need a hash map (`std::unordered_map`) to link pointers with `std::unique_ptr`

Memory Pool Design

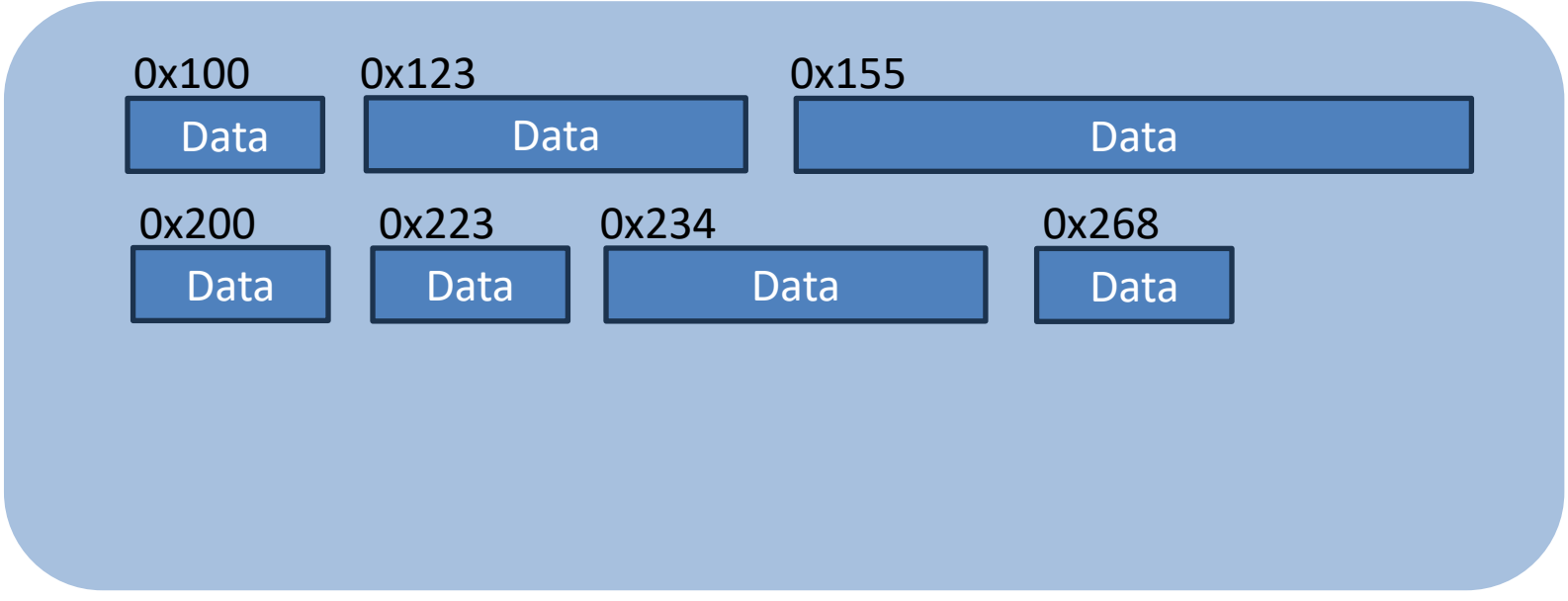
- We thought we need store the pointers to the allocated memory blocks.
  - `std::deque` over `std::vector`
  - Use `std::unique_ptr` for automatic free when the program finish.
- Use a linked list to **store "unused" memory blocks** (pointers)
- We need a hash map (`std::unordered_map`) to link pointers with `std::unique_ptr`
- **~18k** call to alloc (from ~27k) only 1758 real memory allocations.
  - No frees

Memory Pool Design

- Dispose of unnecessary data structures (`deque`, `map`, `unique_ptr`)
  - Only need to keep track of **unused memory blocks**
- On **allocations**, make use of `malloc` (or `new`) and return the **pointer**.
- On **deallocations**, **store** the unused memory blocks

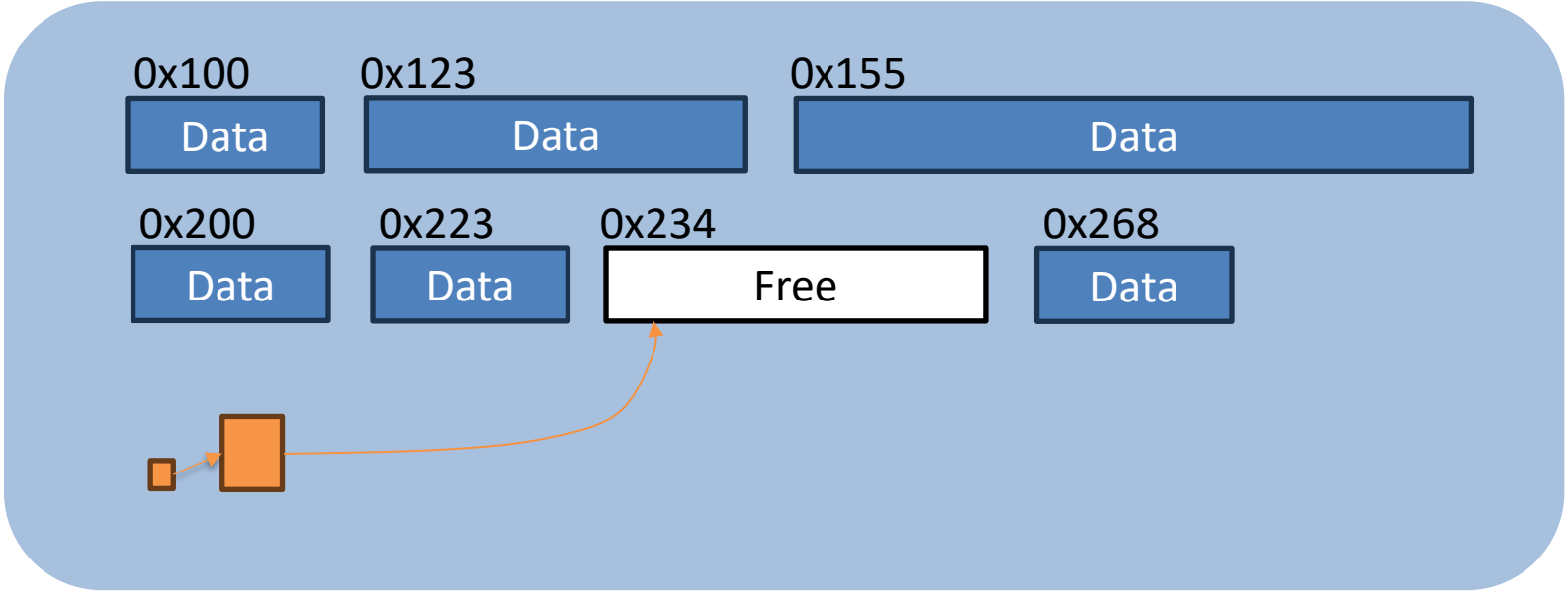
Memory Pool Design

- Improved memory footprint



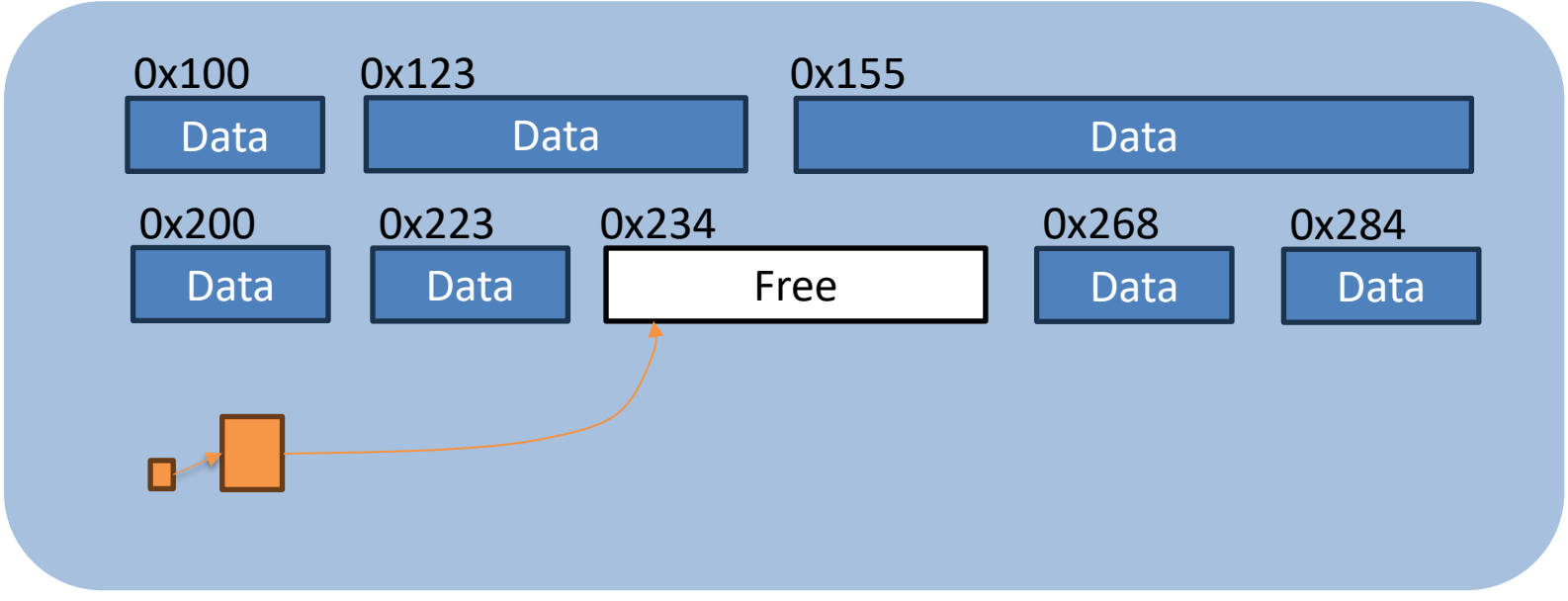
Memory Pool Design

- Improved memory footprint



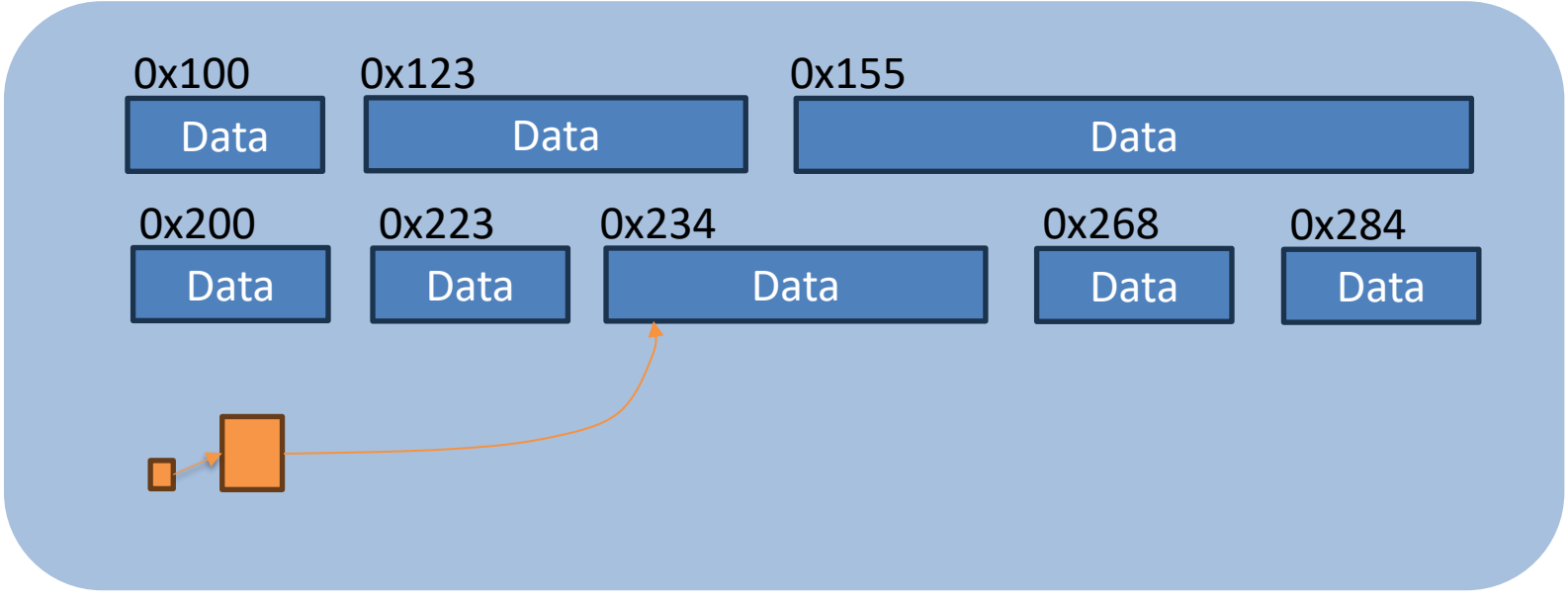
Memory Pool Design

- Improved memory footprint



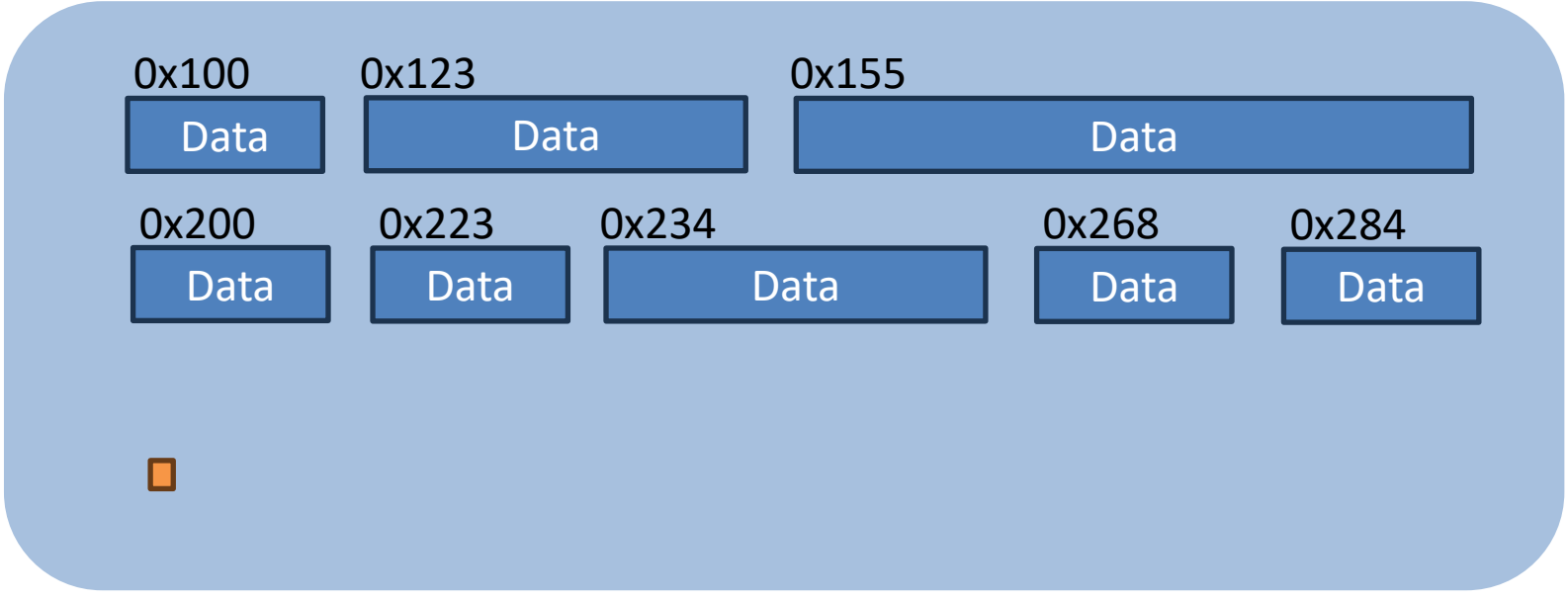
Memory Pool Design

- Improved memory footprint



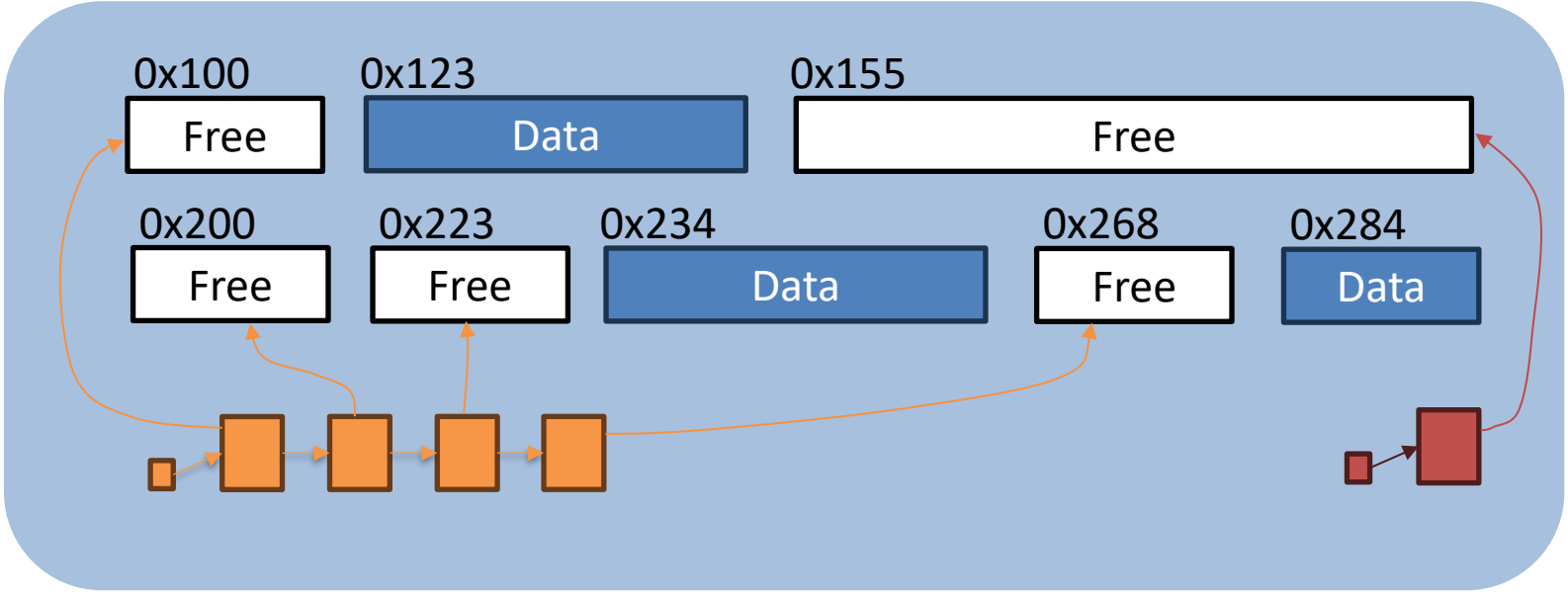
Memory Pool Design

- Improved memory footprint



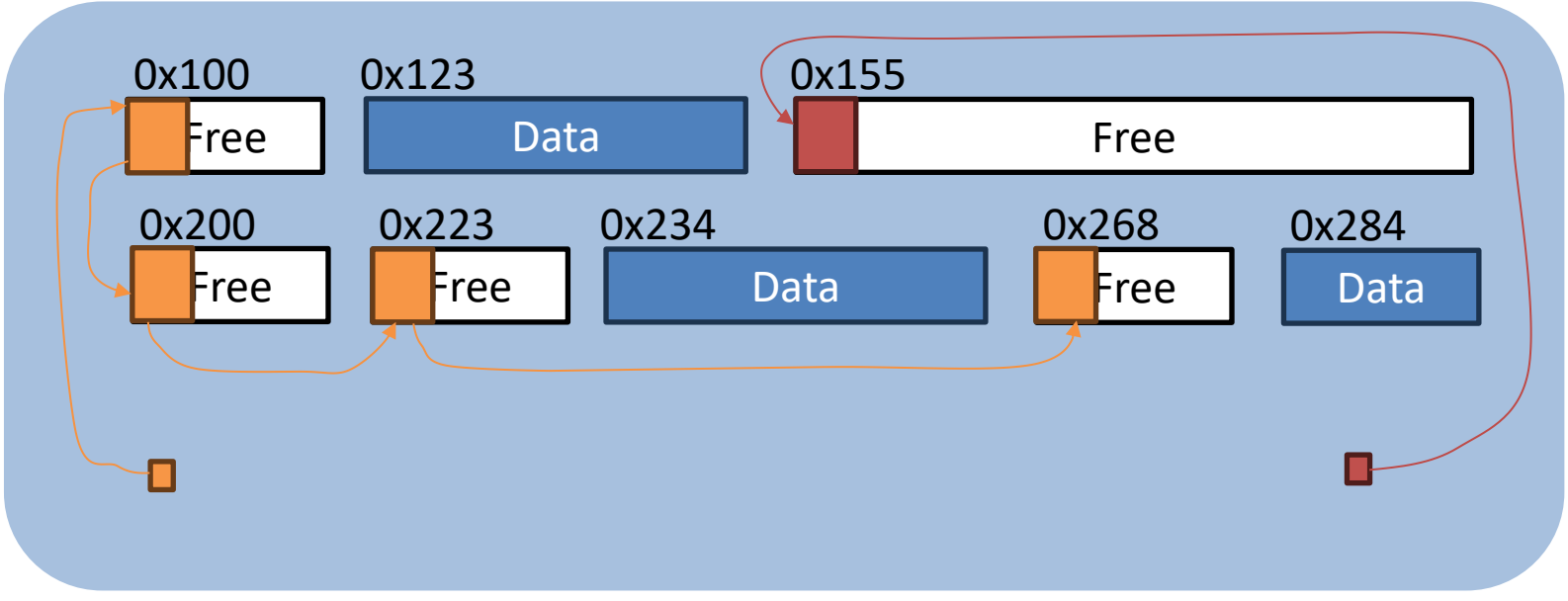
Memory Pool Design

- Our memory footprint



Memory Pool Design

- Our memory footprint



Memory Pool Design

- Dispose of unnecessary data structures (`deque`, `map`, `unique_ptr`)
  - Only need to keep track of **unused memory blocks**
- On **allocations**, make use of `malloc` (or `new`) and return the **pointer**.
- On **deallocations**, **store** the unused memory blocks **in the blocks themselves**.
- To **reduce** the number of "unused" lists, we only create **power-of-two** blocks of memory.

AGENDA

1. Motivation
2. Open Foam Memory Usage
3. Memory Pool Design
4. **Experimental Results**
5. Conclusions and Future Work

Experimental results

- Per iteration: ~18k allocations (of ~27k)
  - 1758 are actually allocations
  - ~16k are reused memory
  - 0 of ~27k frees
- **Highly efficient** allocator, after first iteration is an access to a pointer.
- We did **not** achieve a clear **improvement**.
- We found some improvement in specific cases for **certain solvers**.

AGENDA

1. Motivation
2. Open Foam Memory Usage
3. Memory Pool Design
4. Experimental Results
5. **Conclusions and Future Work**

Conclusions and Future Work

- **Improving** the memory usage is a very complex problem.
- **Memory Allocations** in simplified programs do not reflect their cost. The overlap of the reserves with the computation must be taken into account.
- **Work in progress**: more to learn about solvers memory usage.
- The use of the memory pool could help optimize data layout, leading to improved performance.

The Impact of Memory Management on OpenFoam Performance

Thank you

david.concha@urjc.es

iPC
INSTITUTE FOR
POLYMERS AND COMPOSITES

CAPO
COMPUTER VISION

Universidad
Rey Juan Carlos