Composition in EbbRT

Dan Schatzberg, Boston University

Objective: Build more efficient software by constructing custom, application-specific operating systems.

Objective: Build more efficient software by constructing custom, application-specific operating systems.

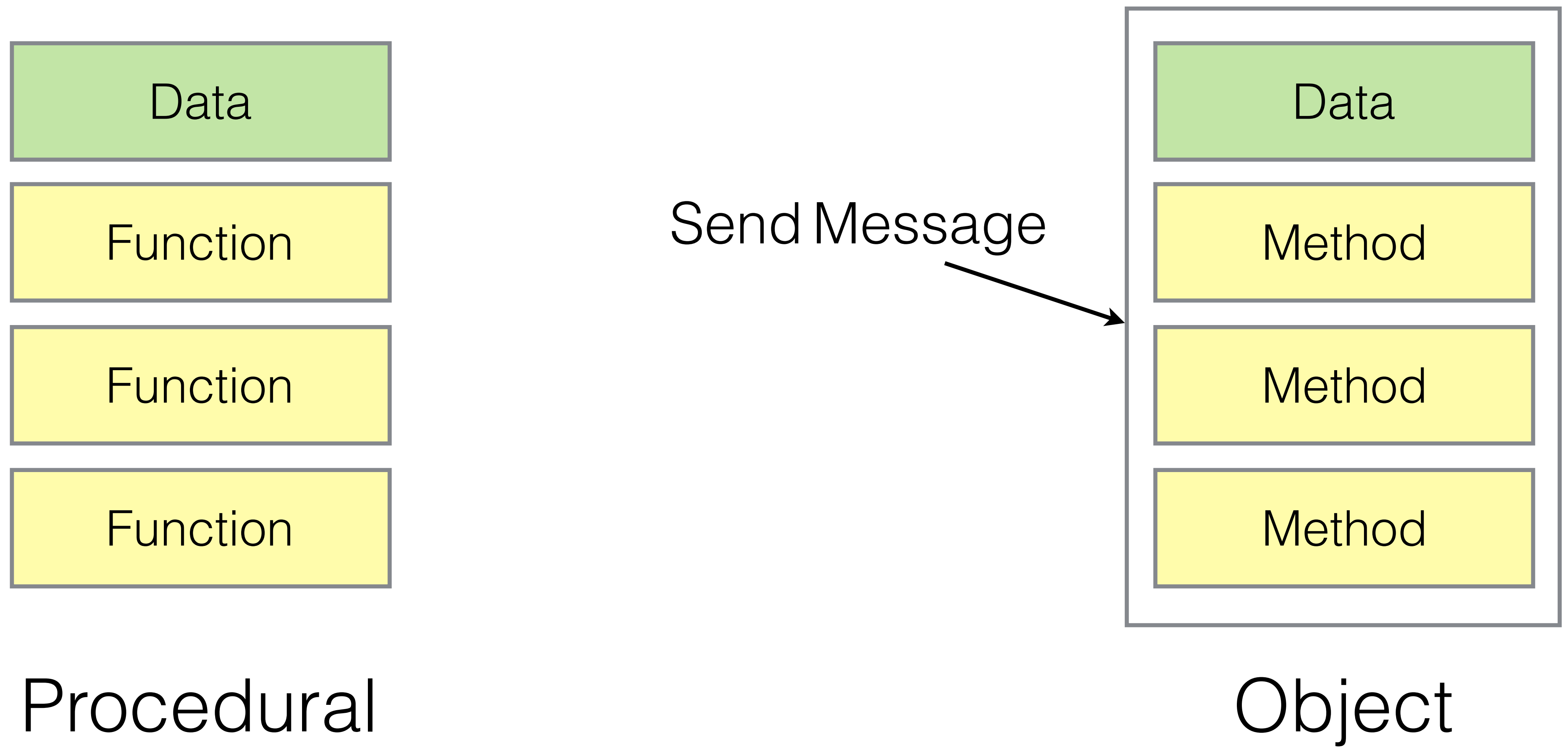
How do we balance the desire to customize with the need to make development scale?

- Integration with general purpose systems for incremental development
- Event-driven, non-preemptive execution model maps software closely to hardware
- Reusable software components which developers can extend, replace, or discard to construct custom systems

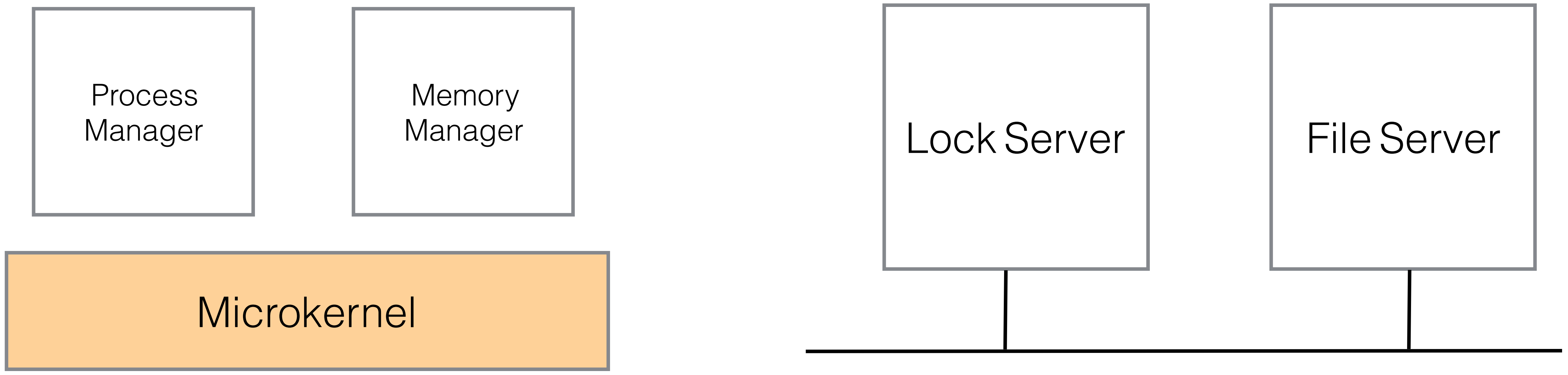
- EbbRT is a toolkit for constructing library operating systems (single address space) for cloud applications

- Components in the small (memory allocators, timers) and in the large (distributed key-value stores, file systems)

Object Oriented Programming (C++, Java)

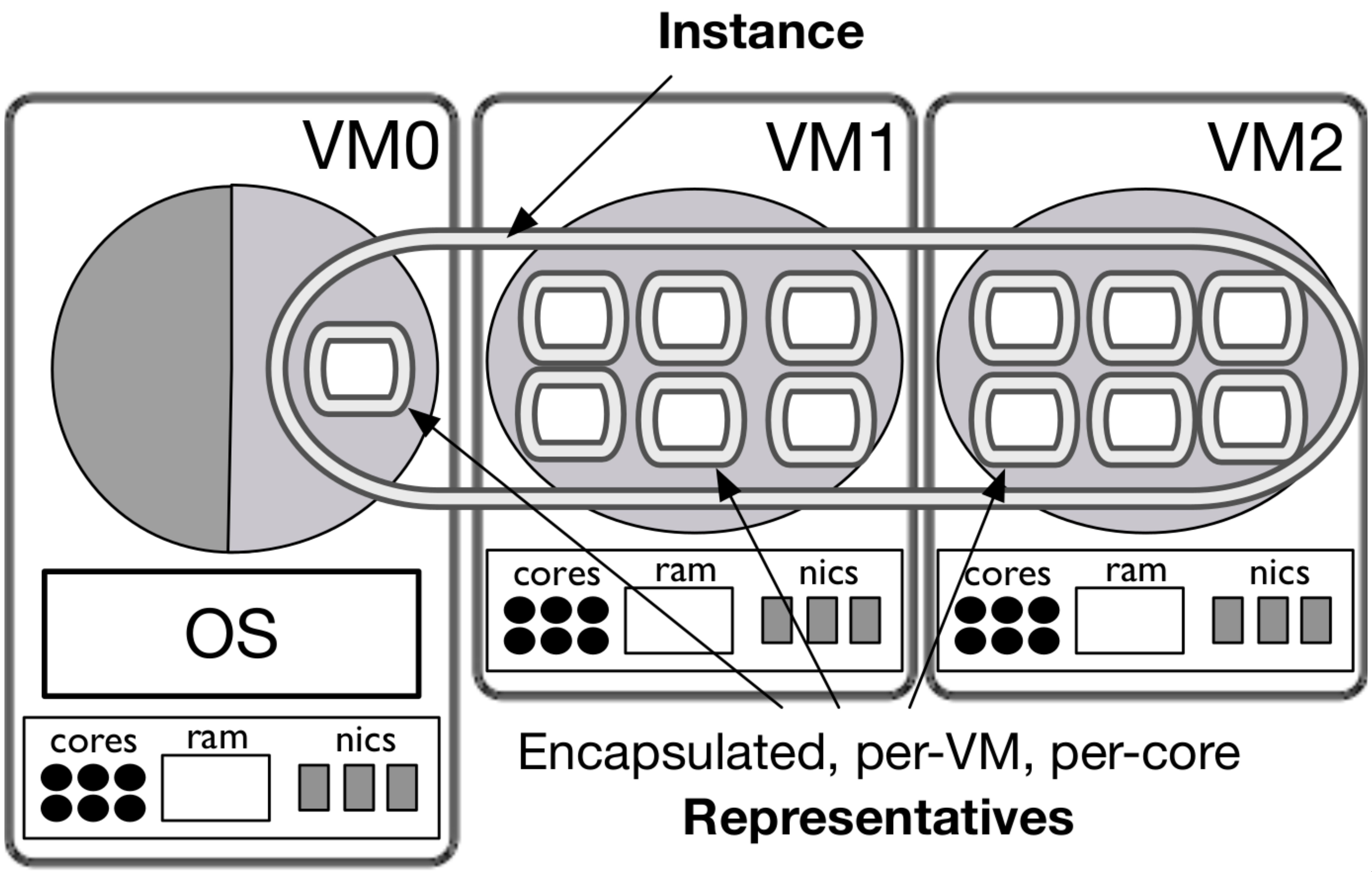


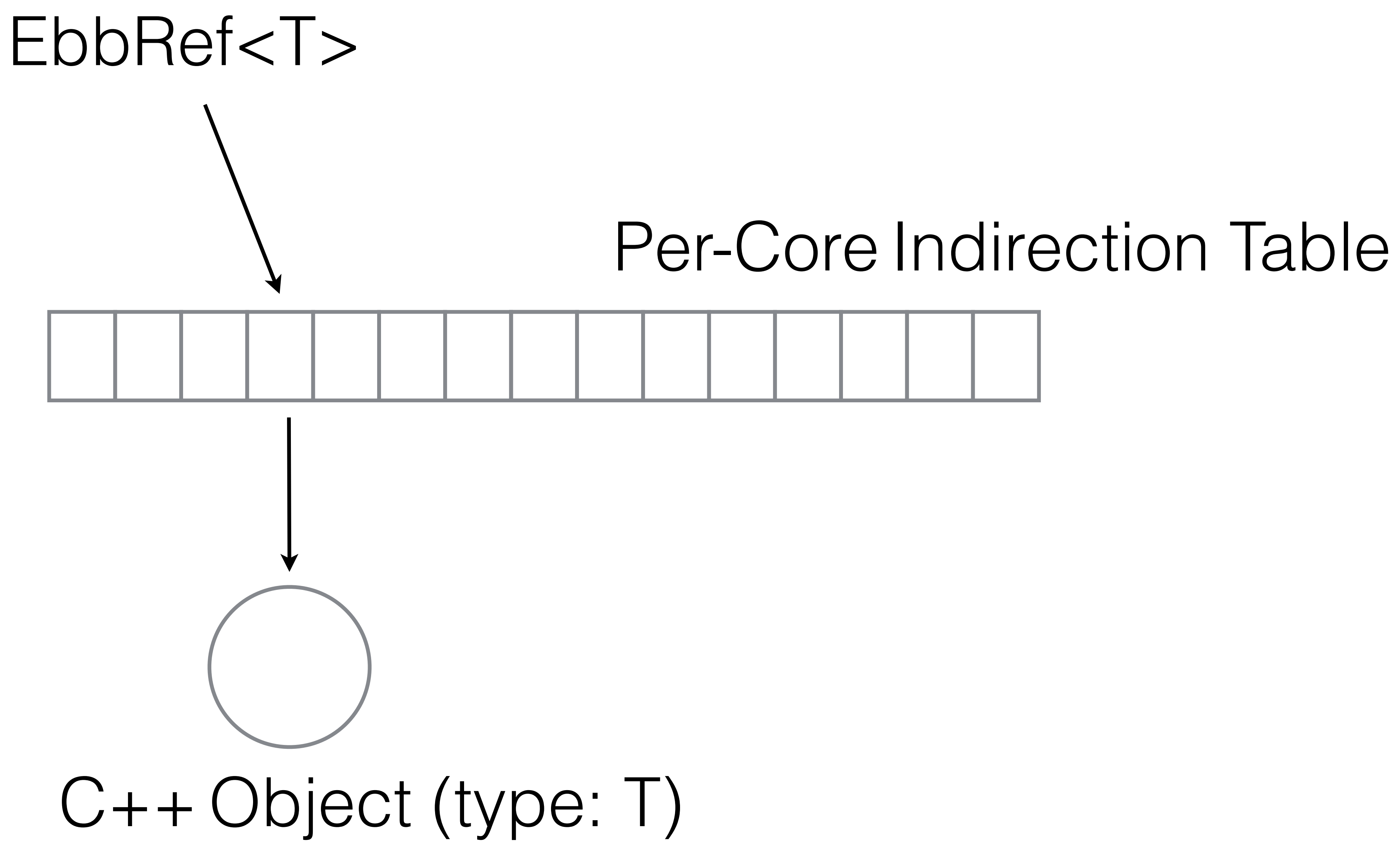
Interoperation without shared memory (CORBA, protobufs, capnproto)

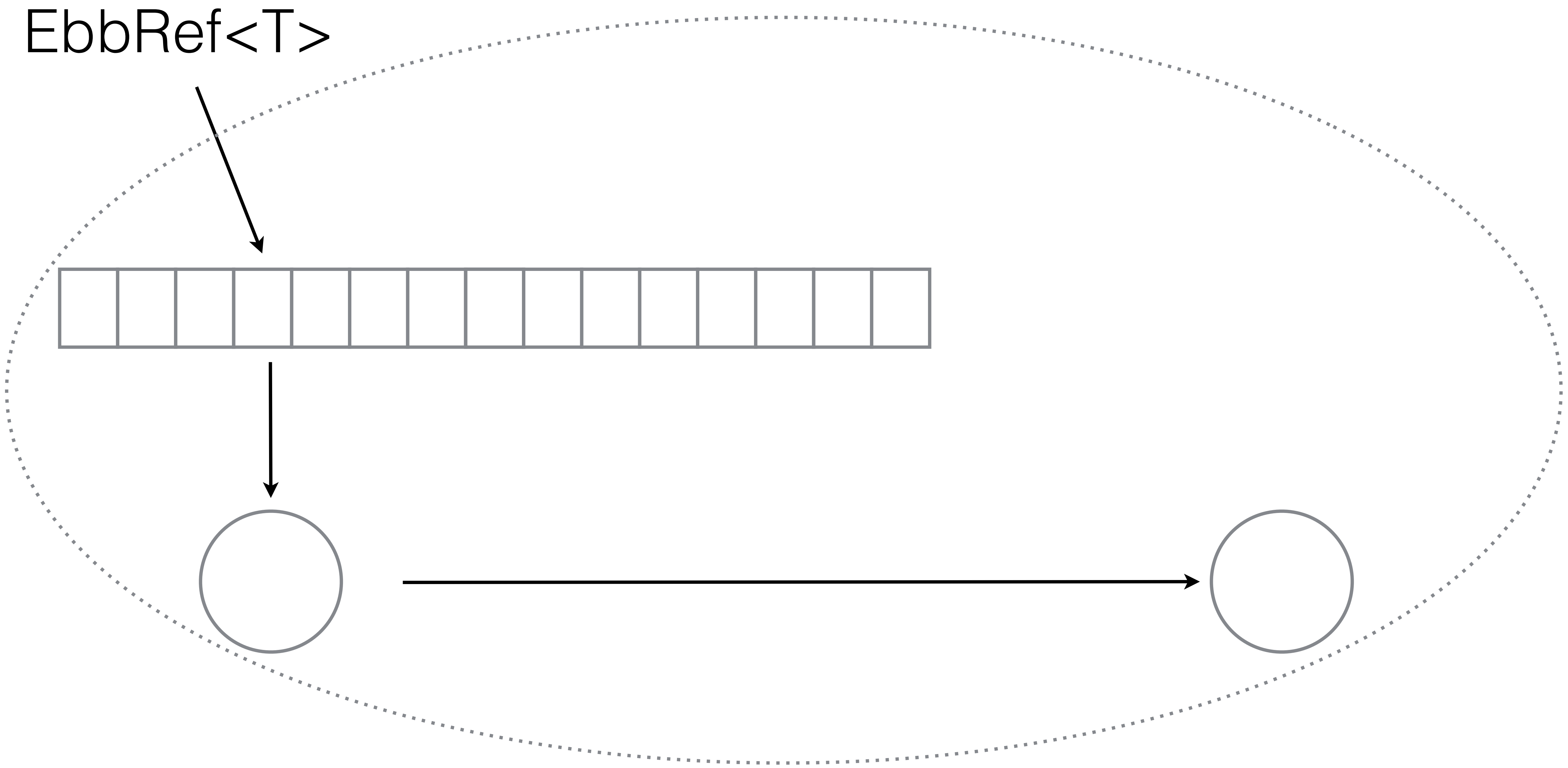


~5% of CPU time in Google datacenters is spent (de)serializing data [ISCA2015]

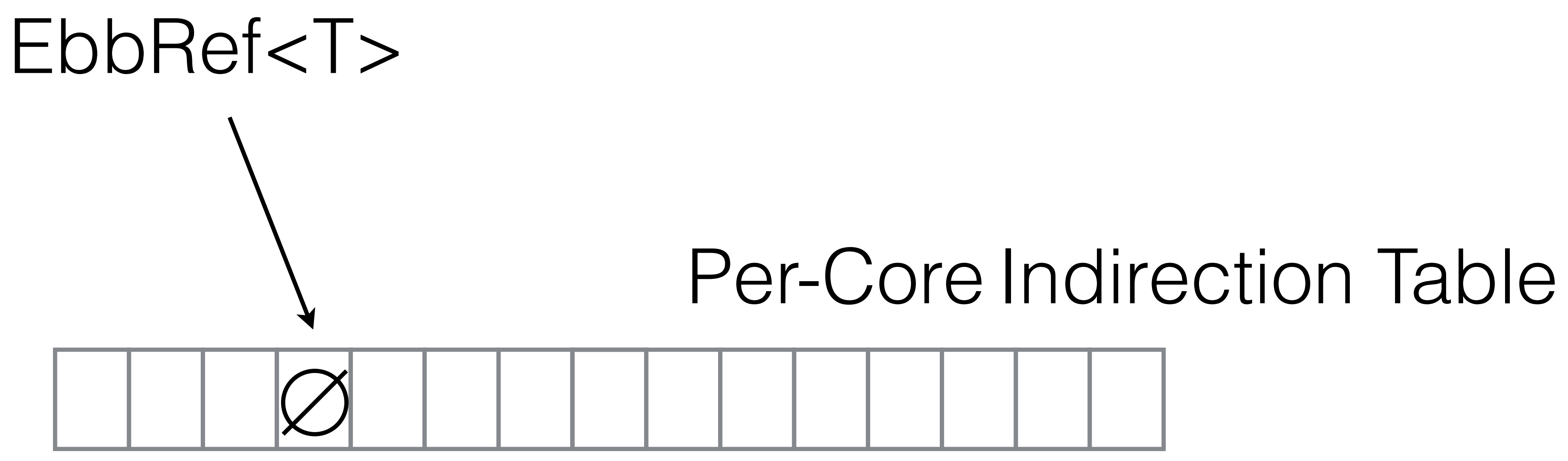
Elastic Building Blocks




EbbRef<T>
Per-Core Indirection Table
C++ Object (type: T)



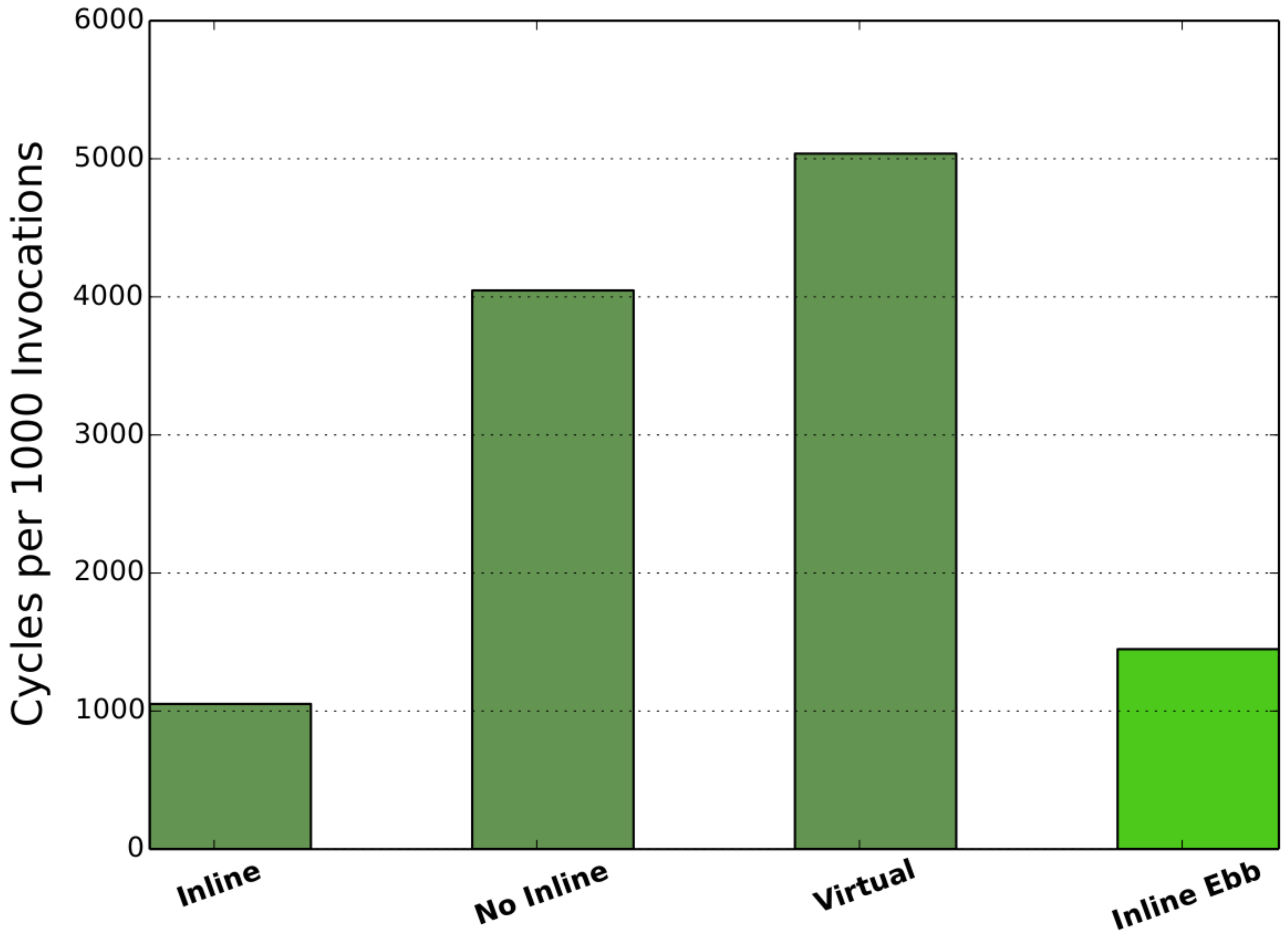
Intra-Ebb Communication is Encapsulated
Free to use shared memory, TCP/IP, RDMA, etc.



Invokes T::HandleFault() to construct a representative

Representatives are constructed on-demand

- Ebbs as services vs data containers. E.g. should a network packet be an Ebb?
- C++ only - how crucial is this?
- Static dispatch (C++ templates) vs Dynamic Dispatch (virtual functions)

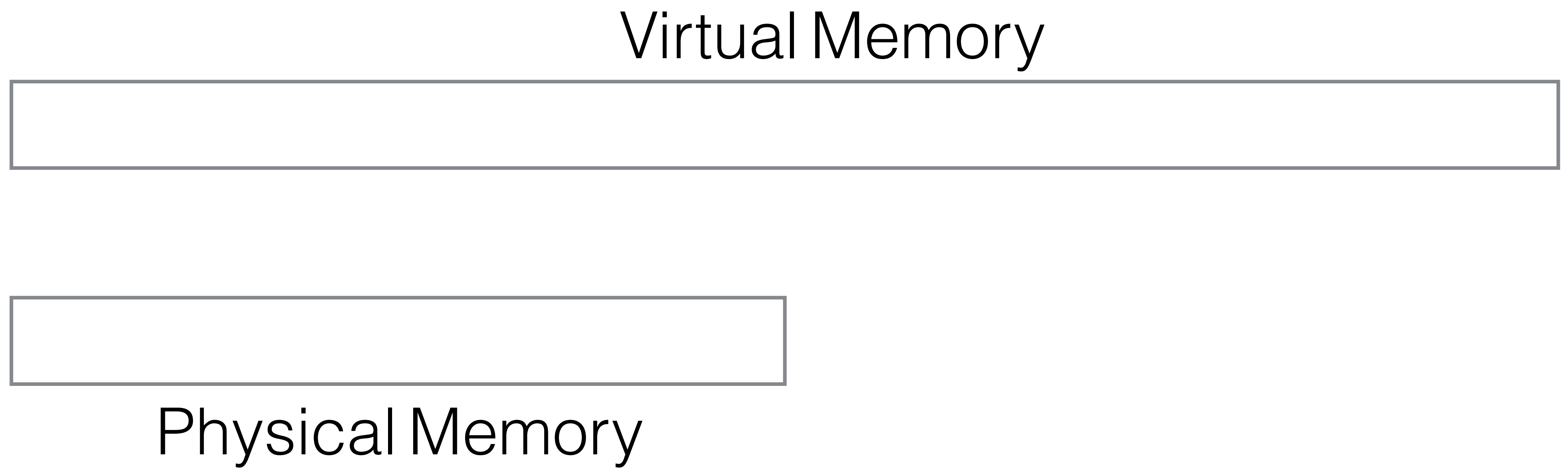


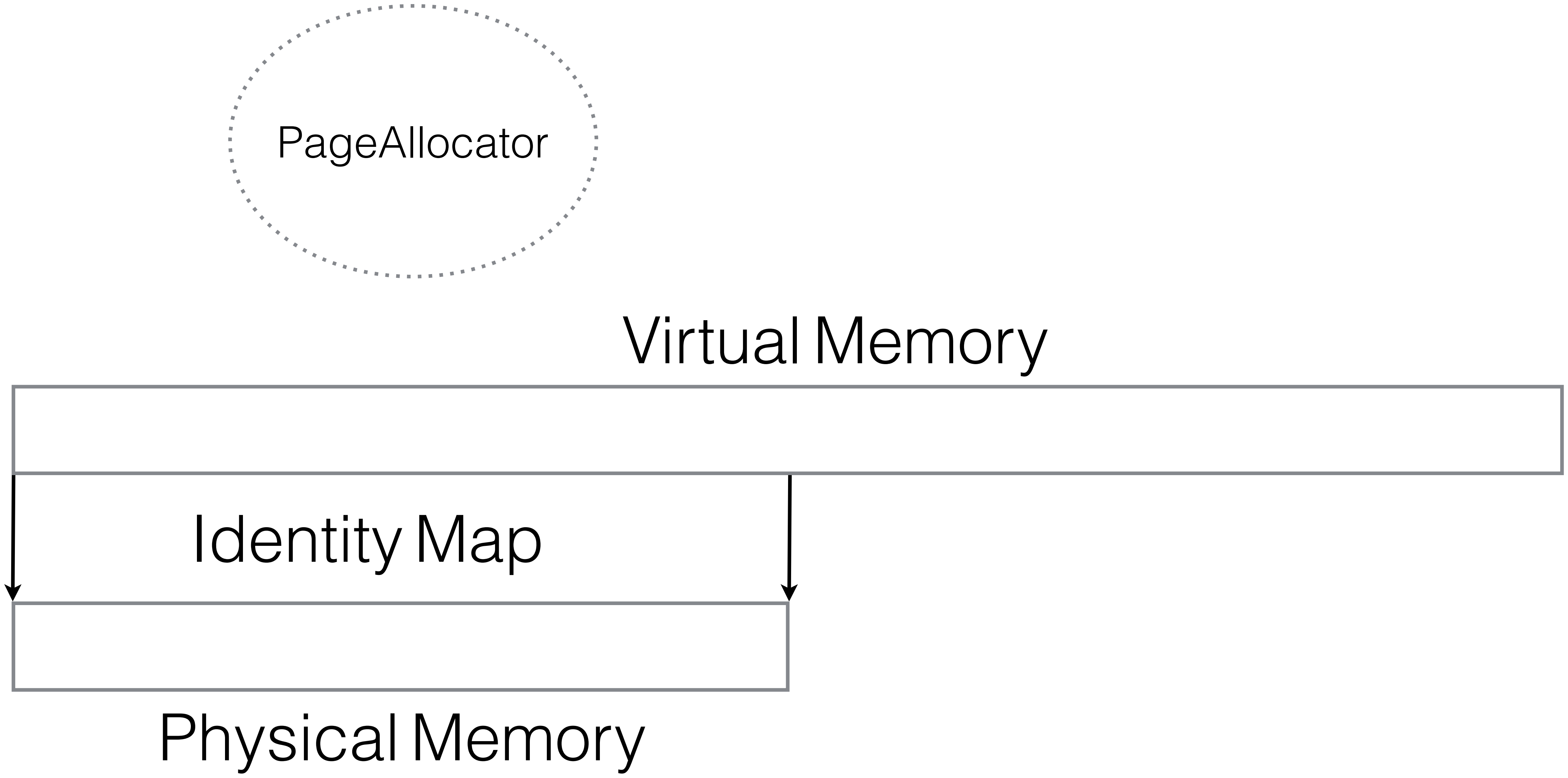
Cycles per 1000 Invocations
6000
5000
4000
3000
2000
1000
0
Inline
No Inline
Virtual
Inline Ebb

- How do we actually define interface semantics?
    - Types
    - Comments
    - Vague Implications

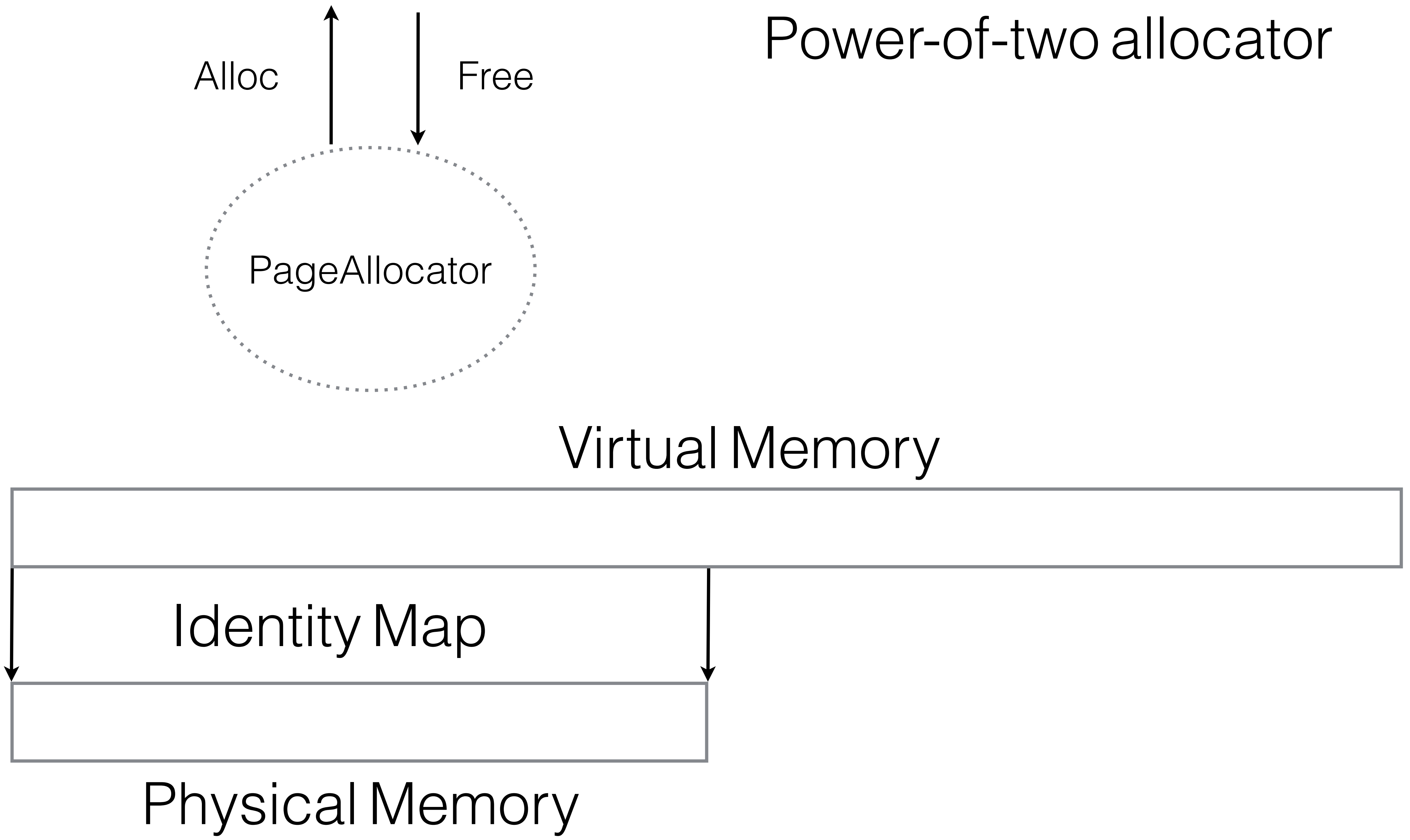
```
Start(std::chrono::microseconds timeout, bool repeat);
```

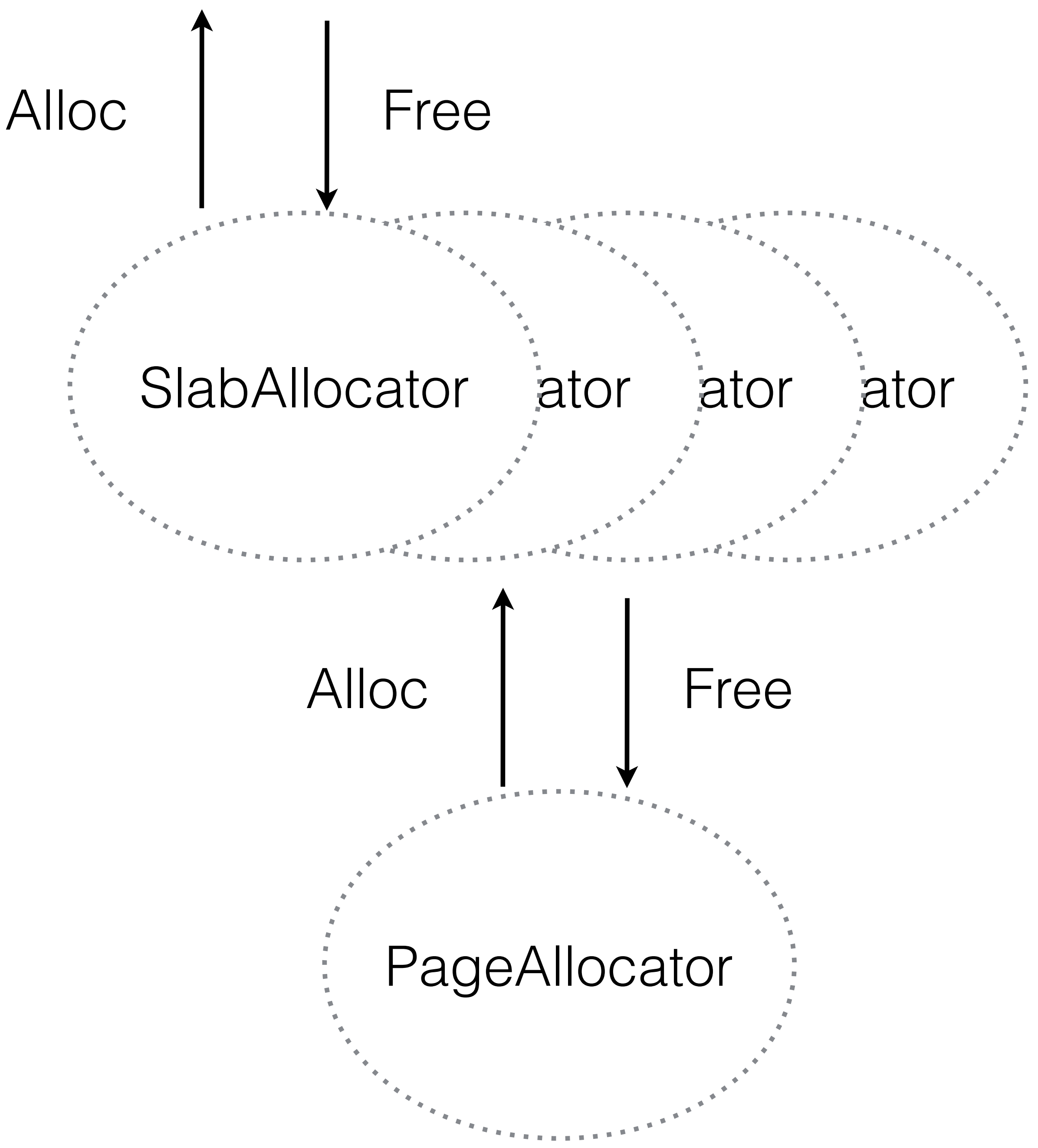
`a_uniq.insert(t)`	`pair<iterator, bool>`	*Requires:* If `t` is a non-const rvalue expression, `value_type` shall be `MoveInsertable` into `X`; otherwise, `value_type` shall be `CopyInsertable` into `X`. *Effects:* Inserts `t` if and only if there is no element in the container with key equivalent to the key of `t`. The `bool` component of the returned pair indicates whether the insertion takes place, and the `iterator` component points to the element with key equivalent to the key of `t`.	Average case $\mathscr{O}(1)$, worst case $\mathscr{O}($`a_uniq.size()`$)$.


Virtual Memory
Physical Memory

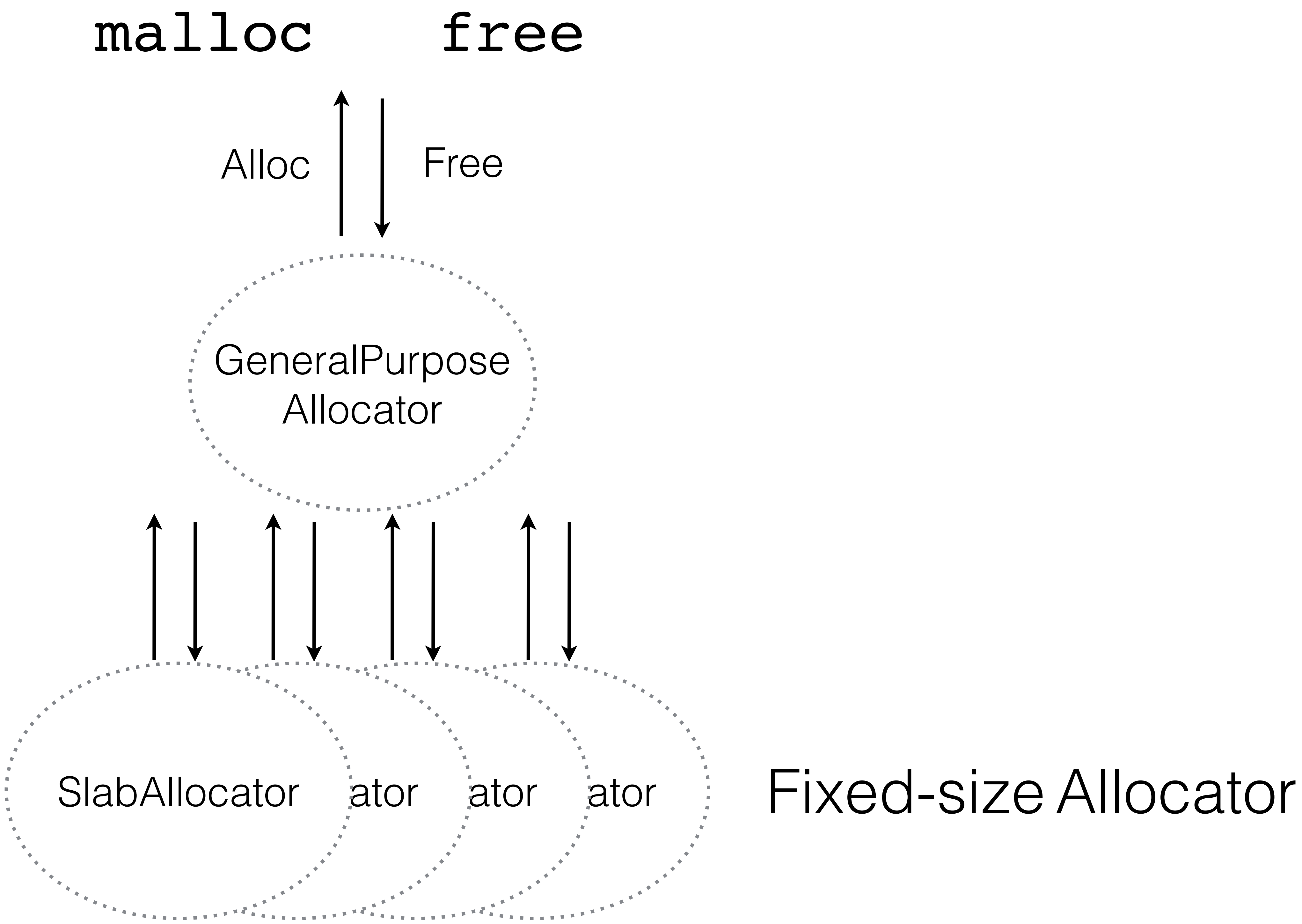

PageAllocator
Virtual Memory
Identity Map
Physical Memory

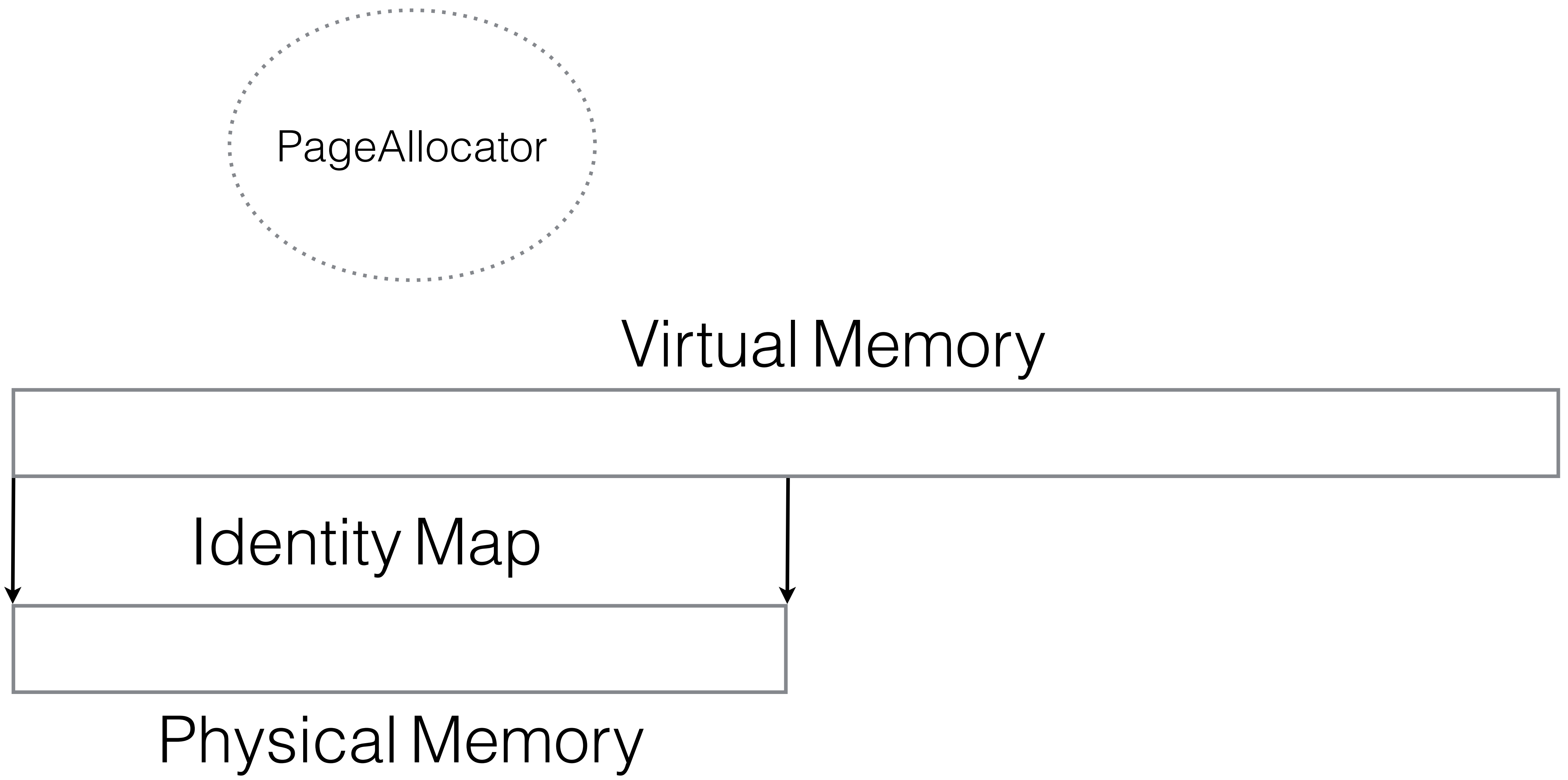
Power-of-two allocator

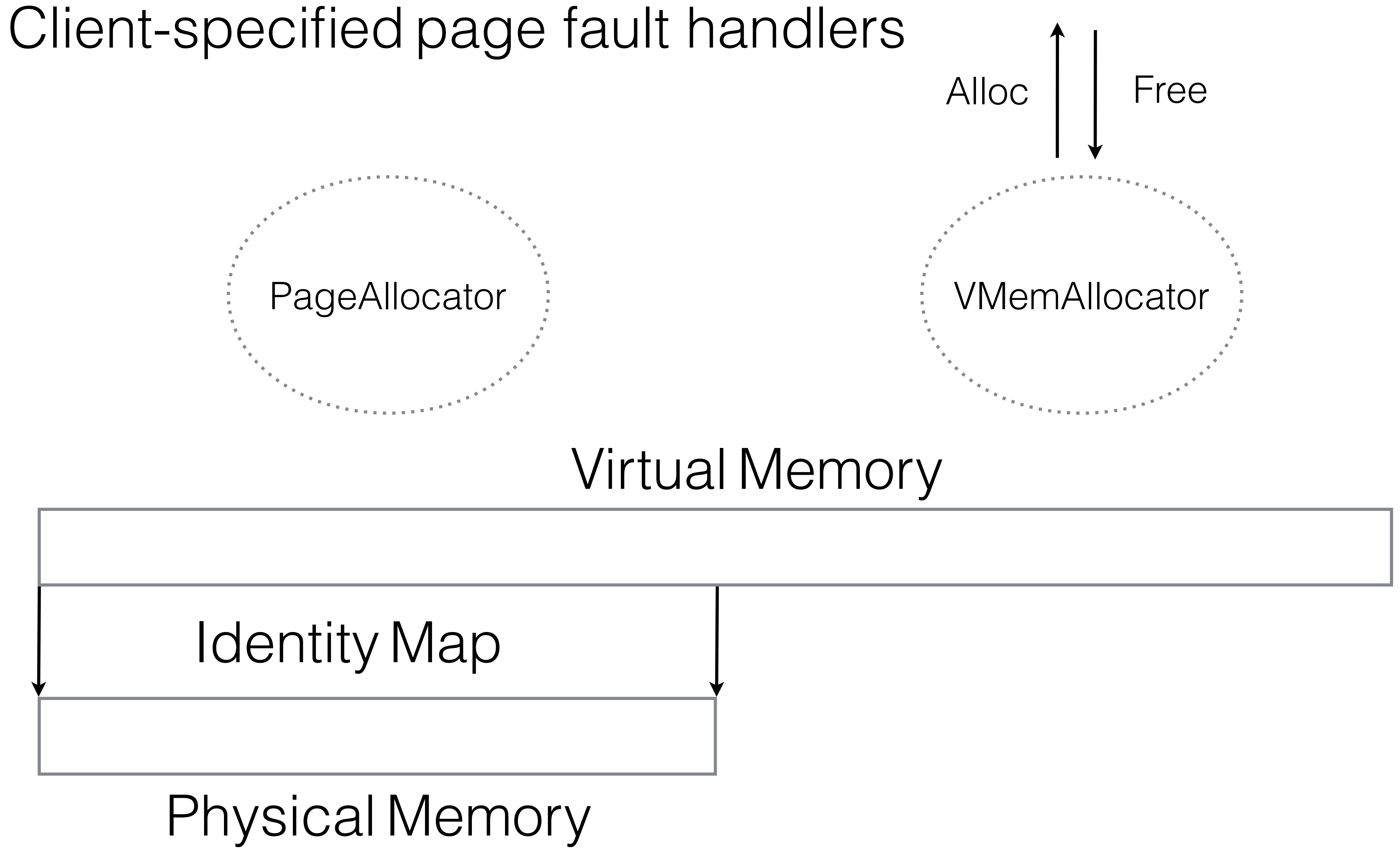


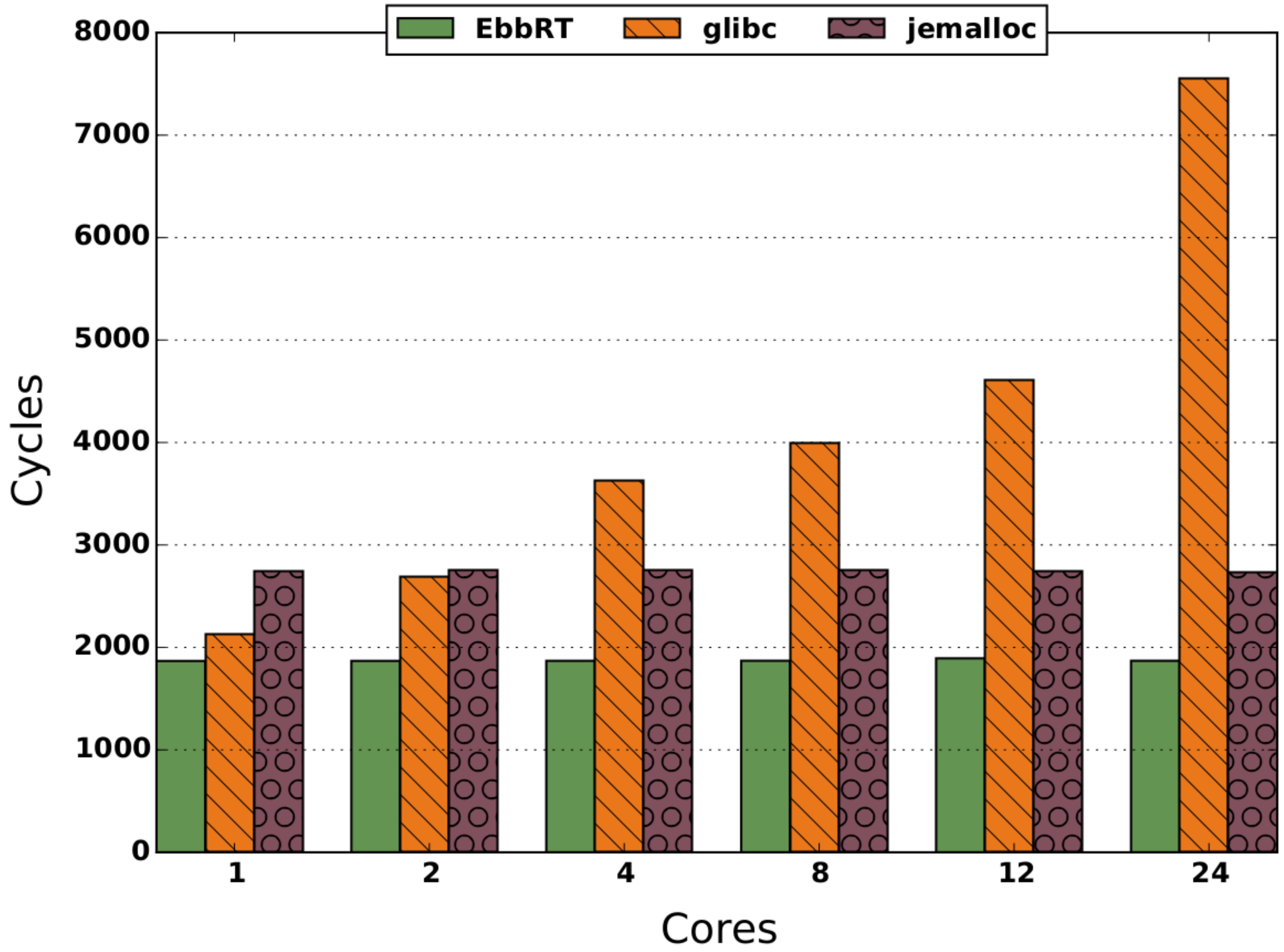


Fixed-size Allocator


malloc
free
Alloc
Free
GeneralPurpose Allocator
SlabAllocator
ator
ator
ator
Fixed-size Allocator


PageAllocator
Virtual Memory
Identity Map
Physical Memory


Client-specified page fault handlers
Alloc
Free
PageAllocator
VMemAllocator
Virtual Memory
Identity Map
Physical Memory


EbbRT
glibc
jemalloc
Cycles
0
1000
2000
3000
4000
5000
6000
7000
8000
Cores
1
2
4
8
12
24

```
movabs 0xffffffff00000010,%rax # EbbRef<GPAllocator>
test   %rax,%rax
je     1870c0 # HandleFault
mov    0x8(%rax),%rdi #Load SlabAllocator Ref
callq  19bd20 <ebbrt::SlabAllocator::Alloc()>
```

https://github.com/sesa/ebbrt

- Memory Allocators (Page, VMem, Slab, GeneralPurpose)
- Networking (Ethernet Driver, IPv4, UDP, DHCP, TCP)
- Filesystem (POSIX: read, write, open, rename, etc.)
- NodeAllocator (Boot (virtual) machine with a particular image, allocate logically isolated networks)
- Messenger (Send messages between Ebb representatives)

- Timer and EventManager (interrupt dispatcher)
- Distributed Key-Value Store (Put(key, value), Get(key))
- Application level data (e.g. Matrix, Image)