CNRS ANF PYTHON

Memory works

Marc Poinot
Numerical Simulation Dept.
marc.poinot@onera.fr

ONERA
THE FRENCH AEROSPACE LAB

retour sur innovation

Outline

- Memory management & Python
  - Basic memory concepts
  - Python
    - allocation concerns
    - reference count
    - numpy
  - workflow and memory ownership

marc.poinot@onera.fr

SCOPE

- Questions
  - What is memory and how does it work with Python?
  - How to share arrays of data from Fortran,C,C++ to Python?
- Answers
  - An overview of who, what, where, when (and maybe why)
  - Simple recipes to make your life more confortable
- Outline
  - Fast survey of memory concepts
  - Memory management with Python/Numpy
  - A strategy for Fortran/C/C++/Python/Numpy assembly

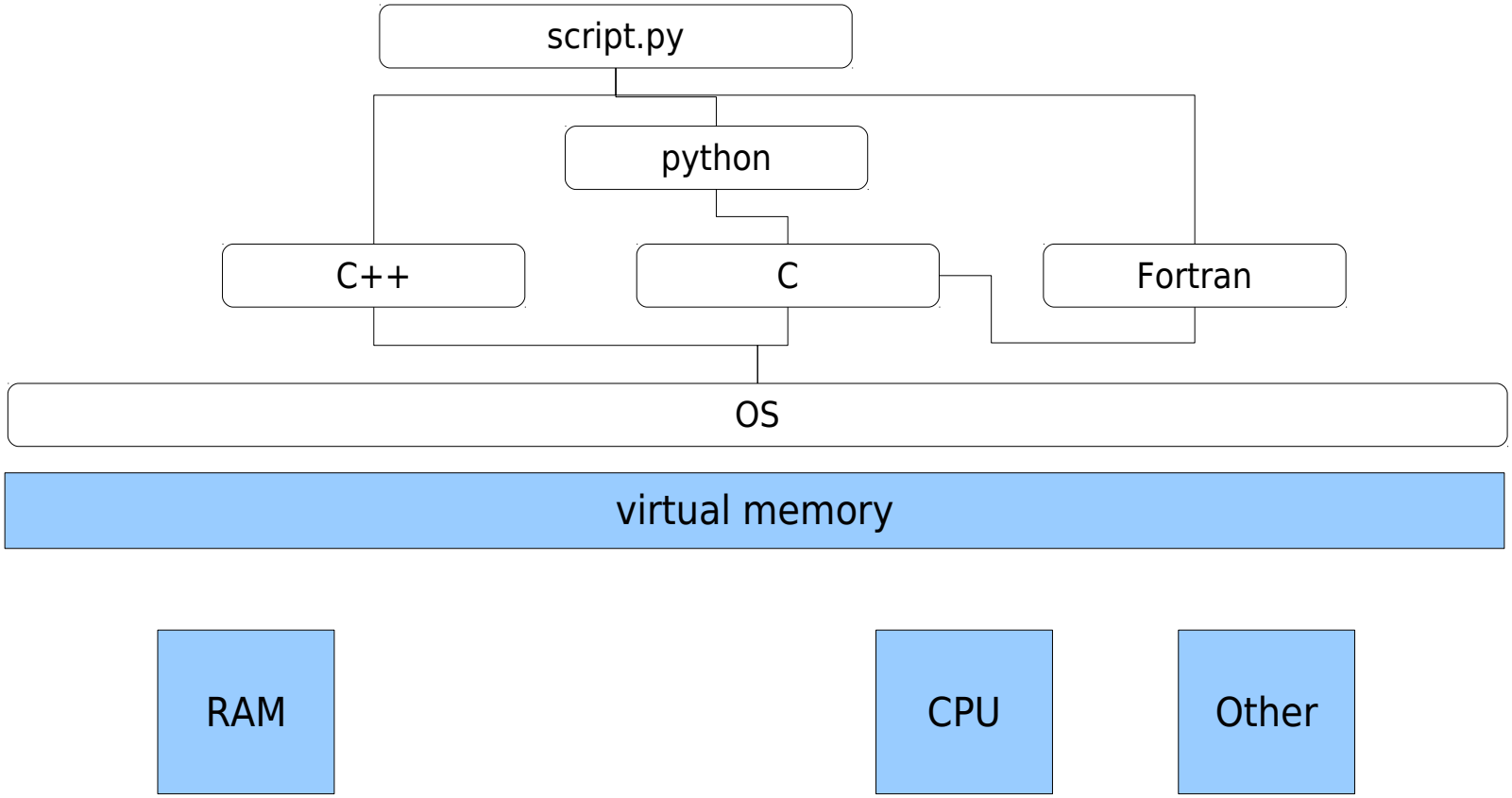
marc.poinot@onera.fr

About memory

- The memory is the part of a computer where you store data
- From the user point of view
  - Store the program and its permanent data
  - Store temporary data and states of the program
  - Exchange data with other programs
- From the computer point of view
  - Physical devices (primary is RAM, secondary is disk)
  - Internal services (caching, paging, segmenting, swapping...)
  - Services for the user (allocate, deallocate, lock...)

Now the computation platforms hardware and operating systems are more and more complex.
We draw here large approximations to make concepts easier to understand, but reality is... complex.

Actors

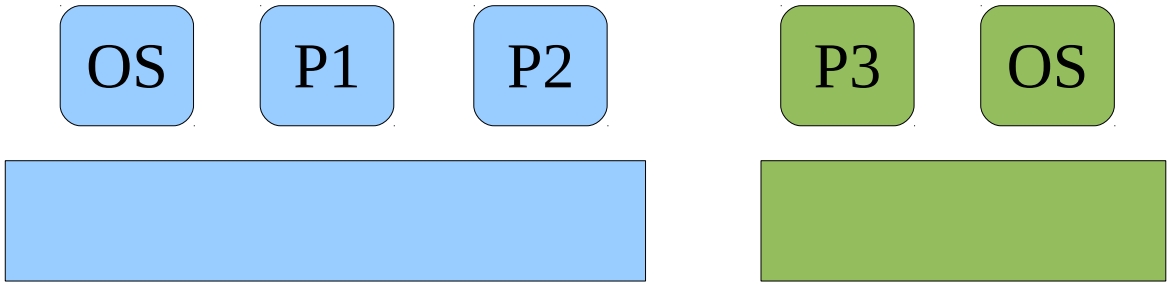


marc.poinot@onera.fr

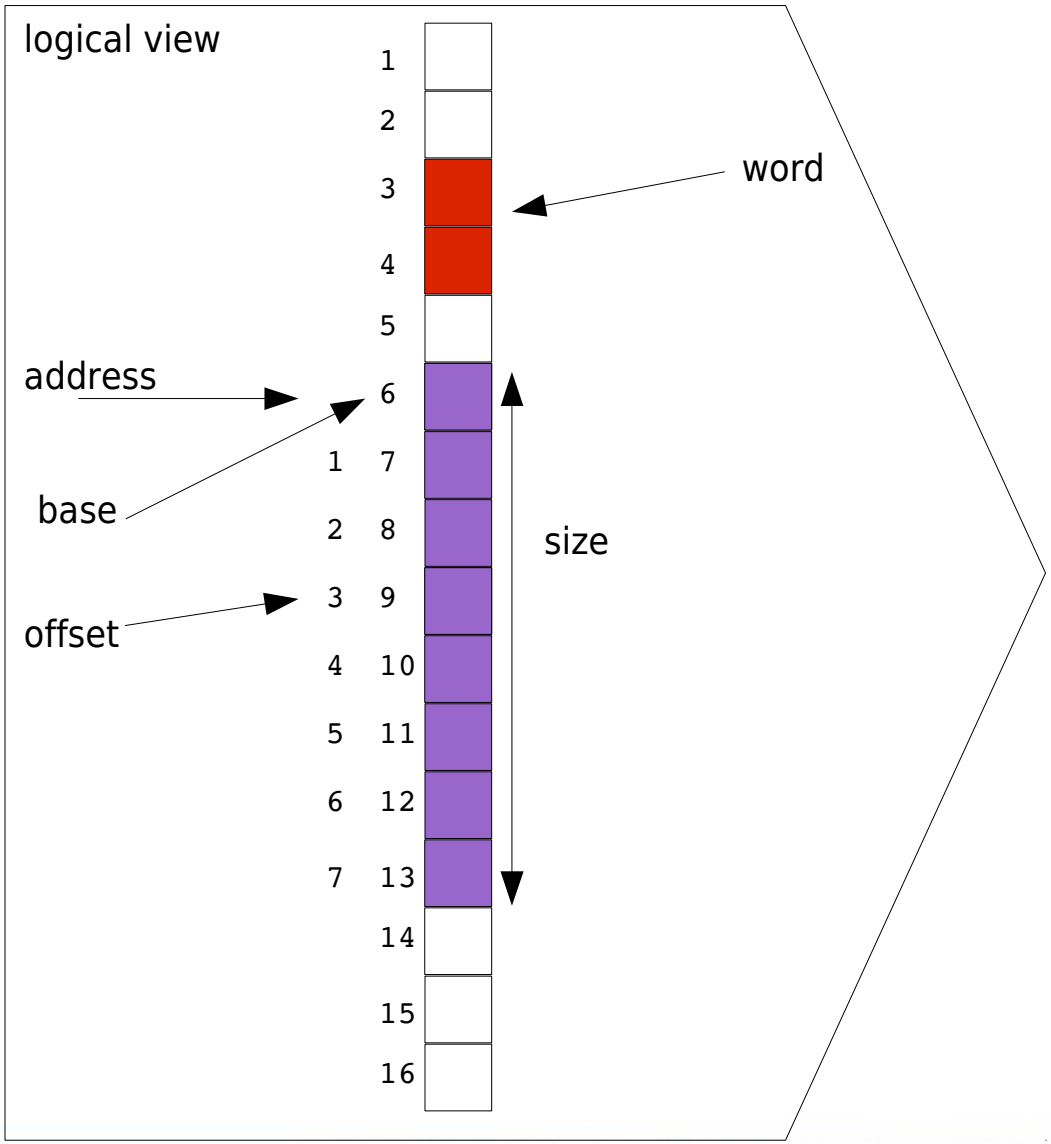
Concurrency

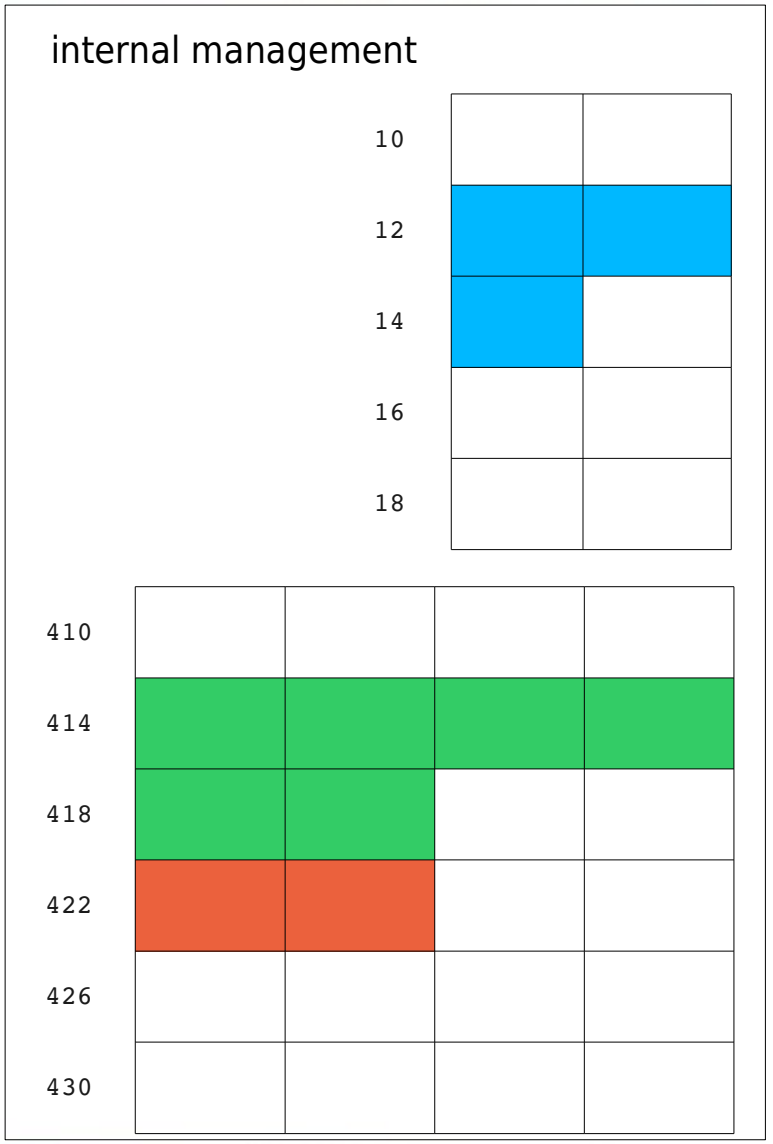
- Memory is allocated for processes
  - The Operating System is your interface to this allocation
  - The OS is a process (more or less one per processor)
  - OS returns an address and reserve the memory up to the size
  - A memory can be reserved for a process
  - A shared memory can be reserved for several process
  - The scope of the address is the OS process itself

- Each time you allocate memory you may stop your process
  - malloc(3) calls brk(2)



Addressing memory

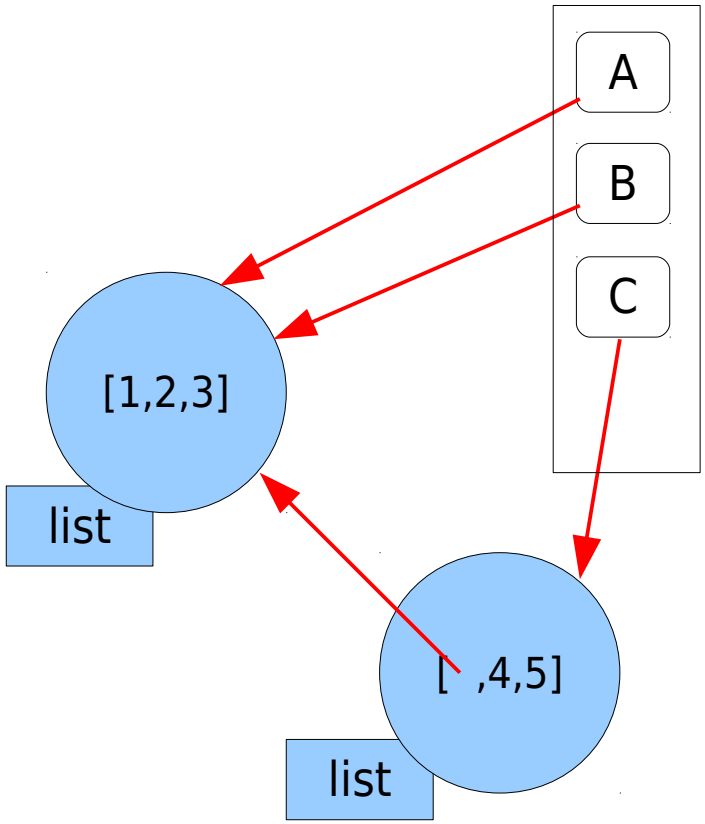




marc.poinot@onera.fr

Reference counting - 1

- Variable/ Object/ Class
  - Each reference to an object is tracked



marc.poinot@onera.fr

Reference counting - 2

```
>>> l1=[1,2,3]
>>> l2=[l1,4,5]
>>> l2
[[1, 2, 3], 4, 5]
>>> l1[2]=7
>>> l2
[[1, 2, 7], 4, 5]
>>> l3=l2[:]
>>> del l1
>>> l1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'l1' is not defined
>>> l2
[[1, 2, 7], 4, 5]
>>> l3
[[1, 2, 7], 4, 5]
>>> l3[2]=0
>>> l3
[[1, 2, 7], 4, 0]
>>> l2
[[1, 2, 7], 4, 5]

>>>
```

Garbage collector

- GC
  - Find objects without reference
  - Release the object memory
  - Not very usefull for memory leak finding

Weak references

- A garbage-able reference
  - Actual reference
  - Not taken into account for refcount
  - Not available on all Python objects

```
>>> import weakref
>>> a=set([1,2,3])
>>> b=weakref.ref(a)
>>> b()
set([1, 2, 3])
>>> a
set([1, 2, 3])
>>> del a
>>> b()
set([1, 2, 3])
>>> b
<weakref at 0x7fad2e777100; dead>
>>> print b()
None
>>>
```

Memory profiling

- External modules
  - memory_profile
    - displays memory use per function
  - objgraph
    - displays relationships between objects

marc.poinot@onera.fr

Memory leaks

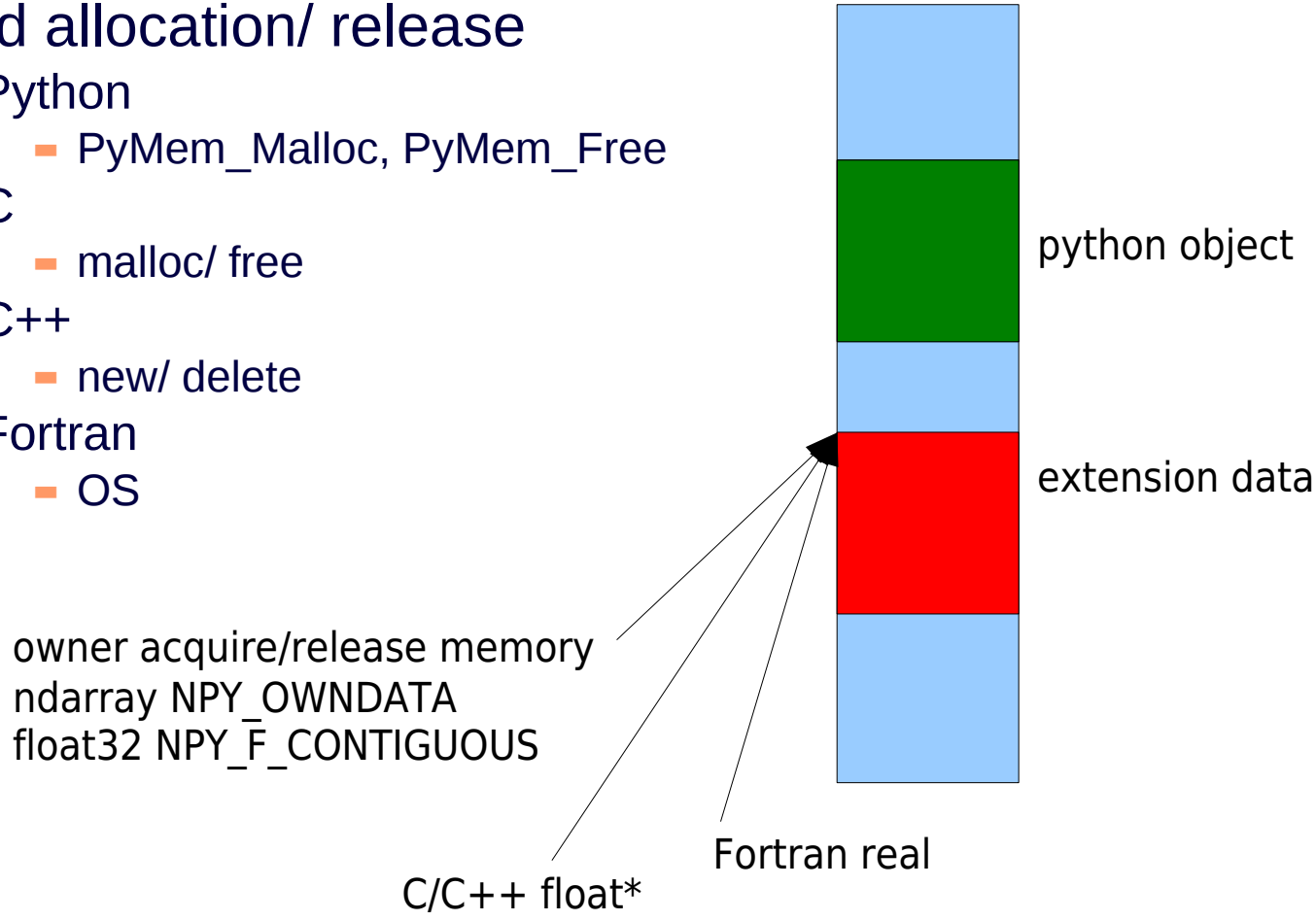
- obmalloc.c
  - Manages arena of fixed size block
  - No way to find back object using this memory
  - No way to move the pointers
  - Arena memory is released only when all objects are released in the arena
  - This can lead to memory leak

Memory leaks

- obmalloc.c
  - Manages arena of fixed size block
  - No way to find back object using this memory
  - No way to move the pointers
  - Arena memory is released only when all objects are released in the arena
  - This can lead to memory leak

Memory ownership

- Mixed allocation/ release
  - Python
    - PyMem_Malloc, PyMem_Free
  - C
    - malloc/ free
  - C++
    - new/ delete
  - Fortran
    - OS



Practical Training

- import sys
- sys.getrefcount(o)
  - Create lists
  - Add references
  - Check counts
  - Use del