I/O Research @ GPPD UFRGS

Francieli Zanon Boito
Rodrigo Virote Kassick

Philippe O. A. Navaux
Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil

Yves Denneulin
INRIA – LIG - University of Grenoble, France

High Performance Computing

Weather forecast, seismic simulations, DNA sequencing, …

… they need to **access and write data to files** that are **shared by all processes**.

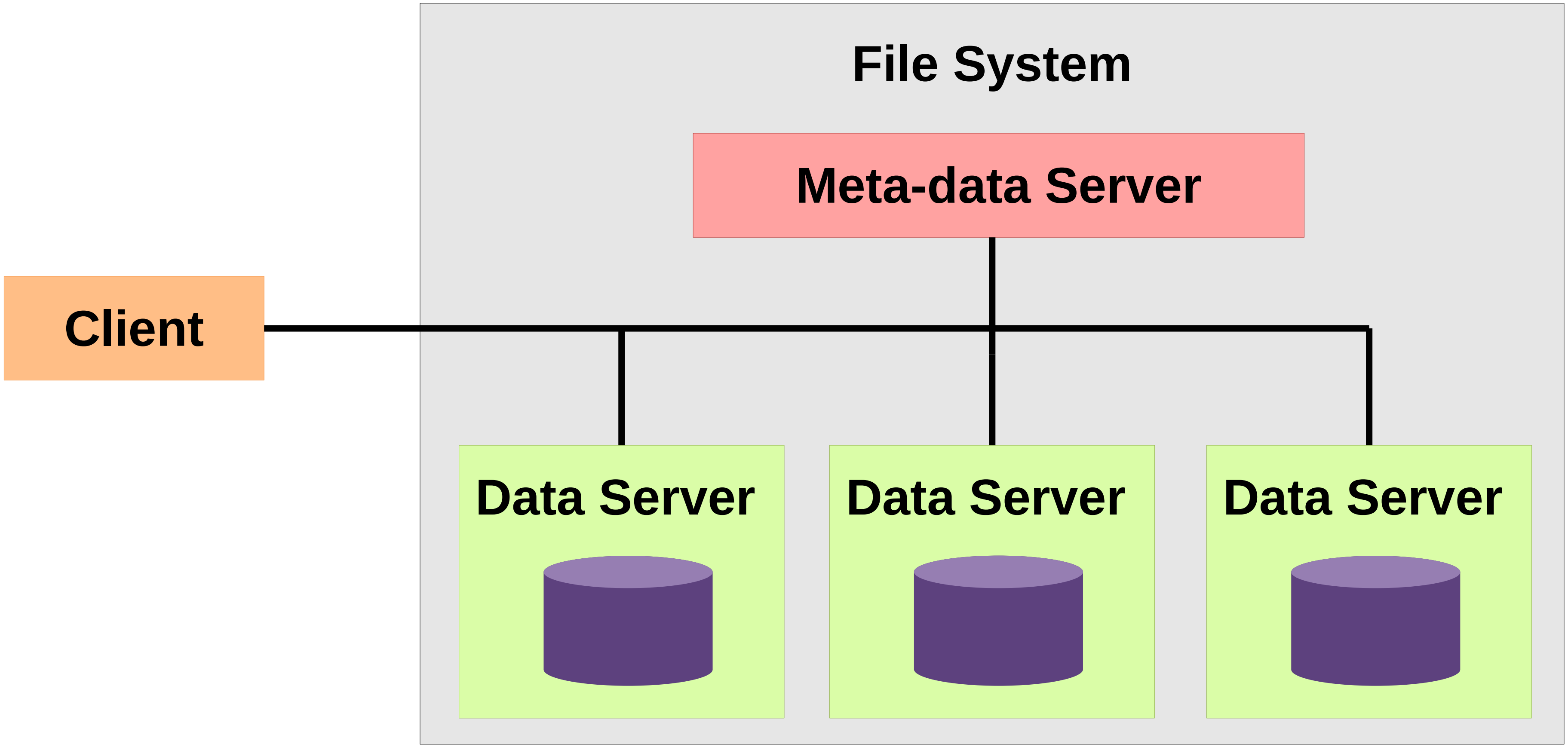
L I G

.inf INSTITUTO DE INFORMÁTICA UFRGS

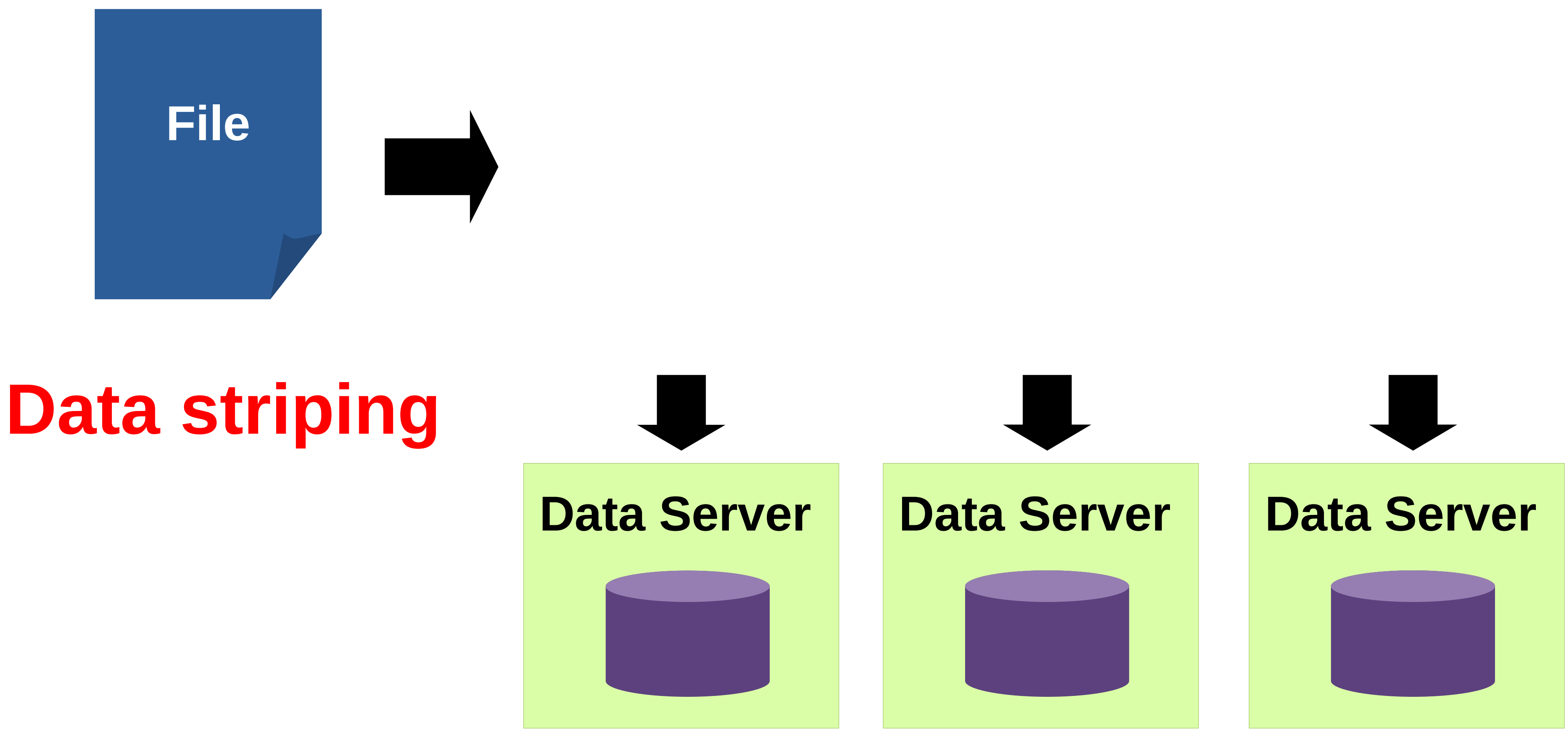
Parallel File Systems

- Allow the access to shared files by all processing nodes
- **Parallel access** to data (focus on **high performance**)
- **Transparent access**: applications do not need to know where the file is
- Examples: **Lustre, PVFS** (OrangeFS)

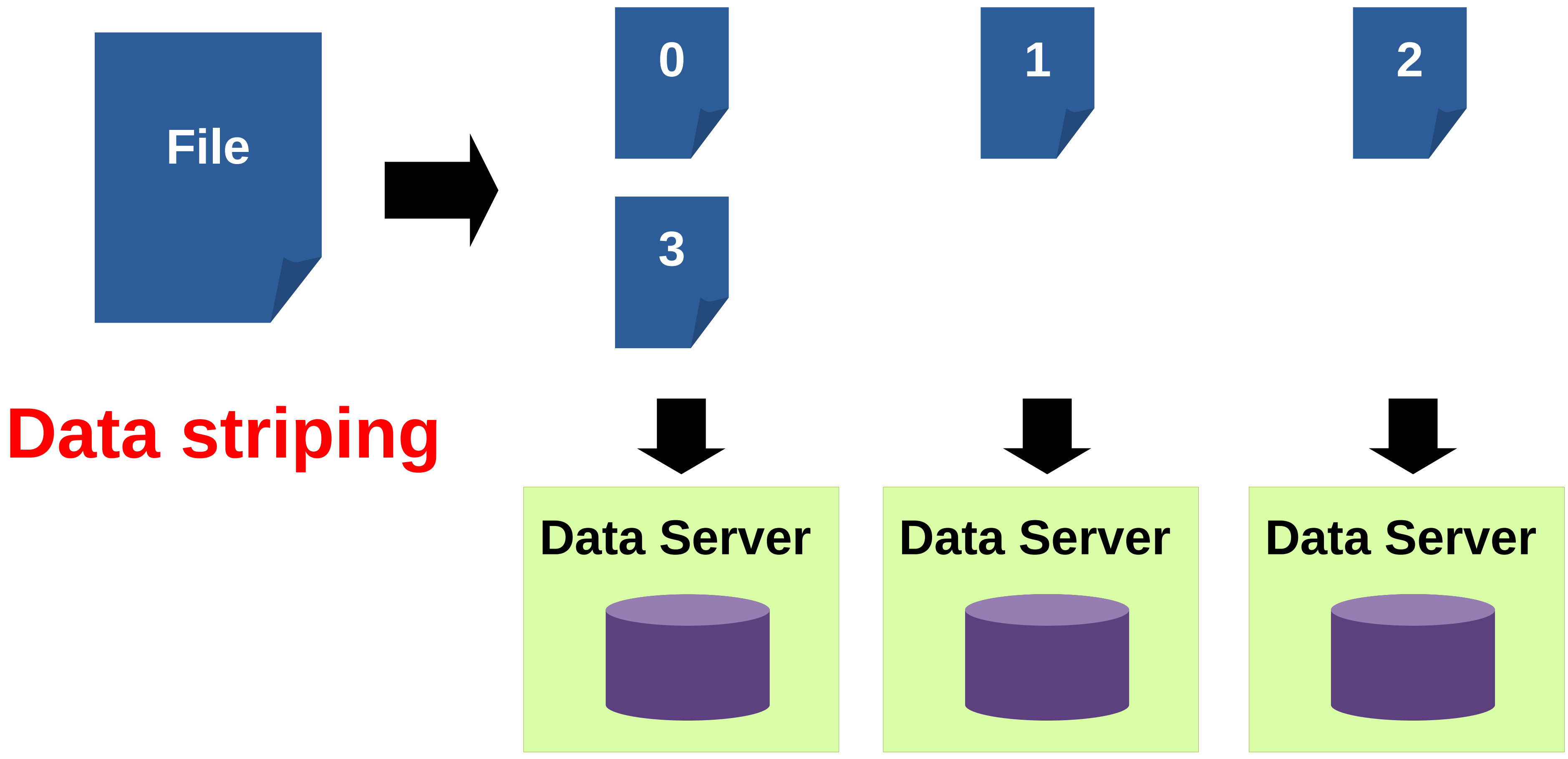
Parallel File Systems



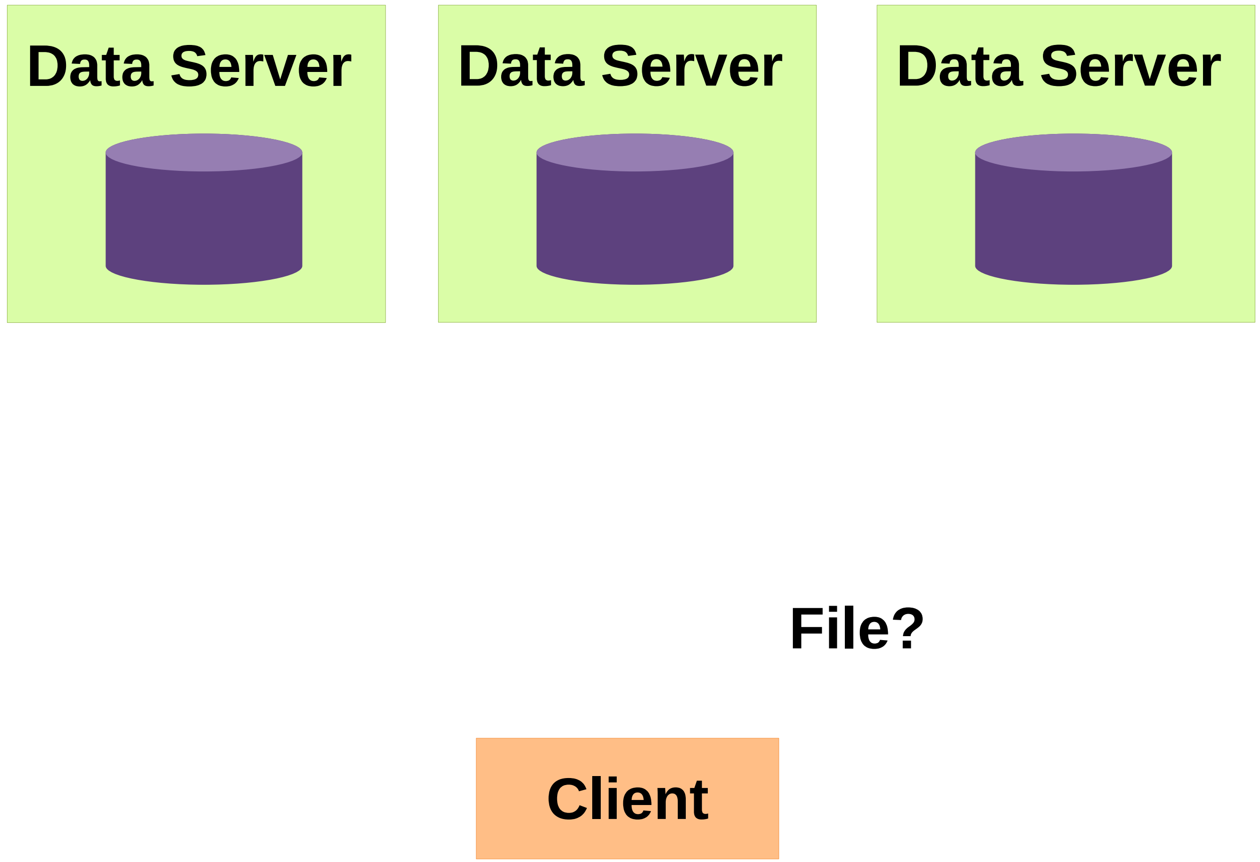
Data Striping



Data Striping

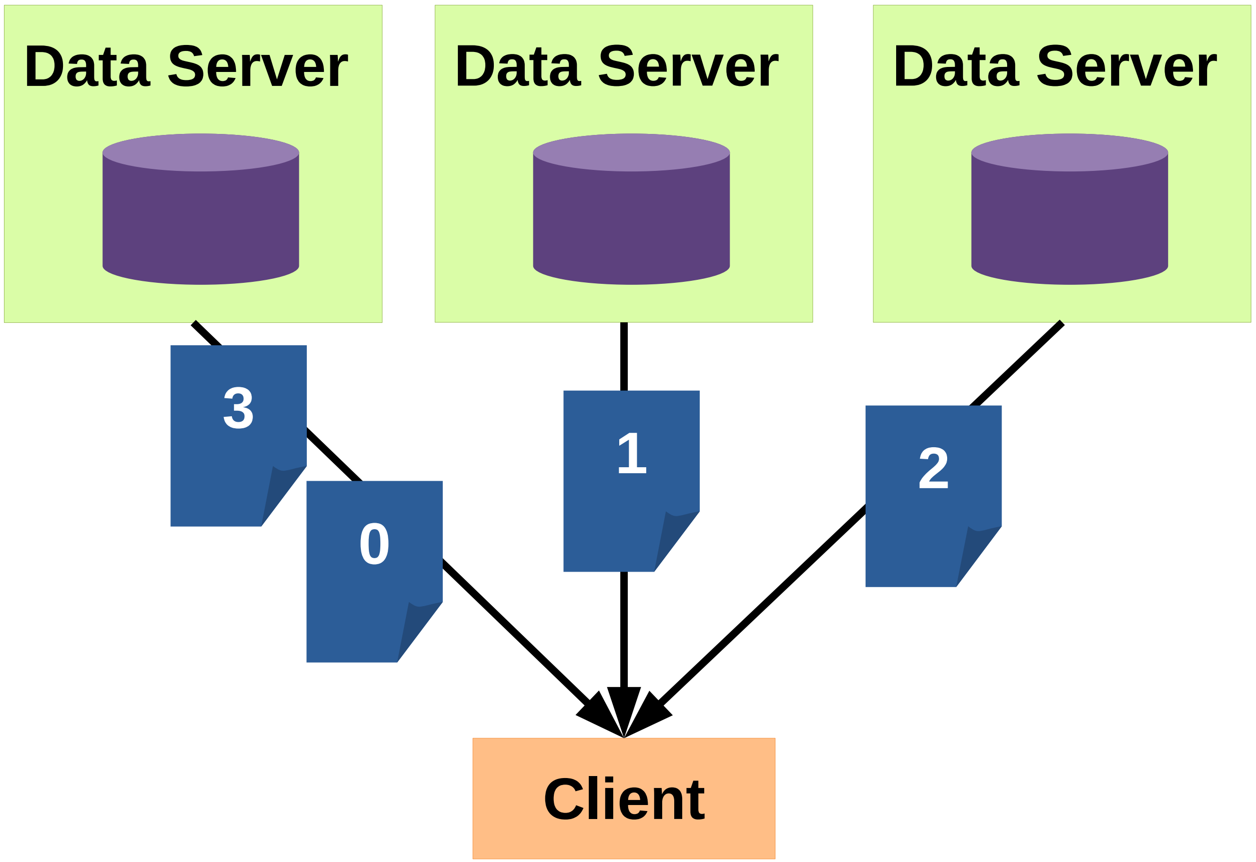


Parallel Access



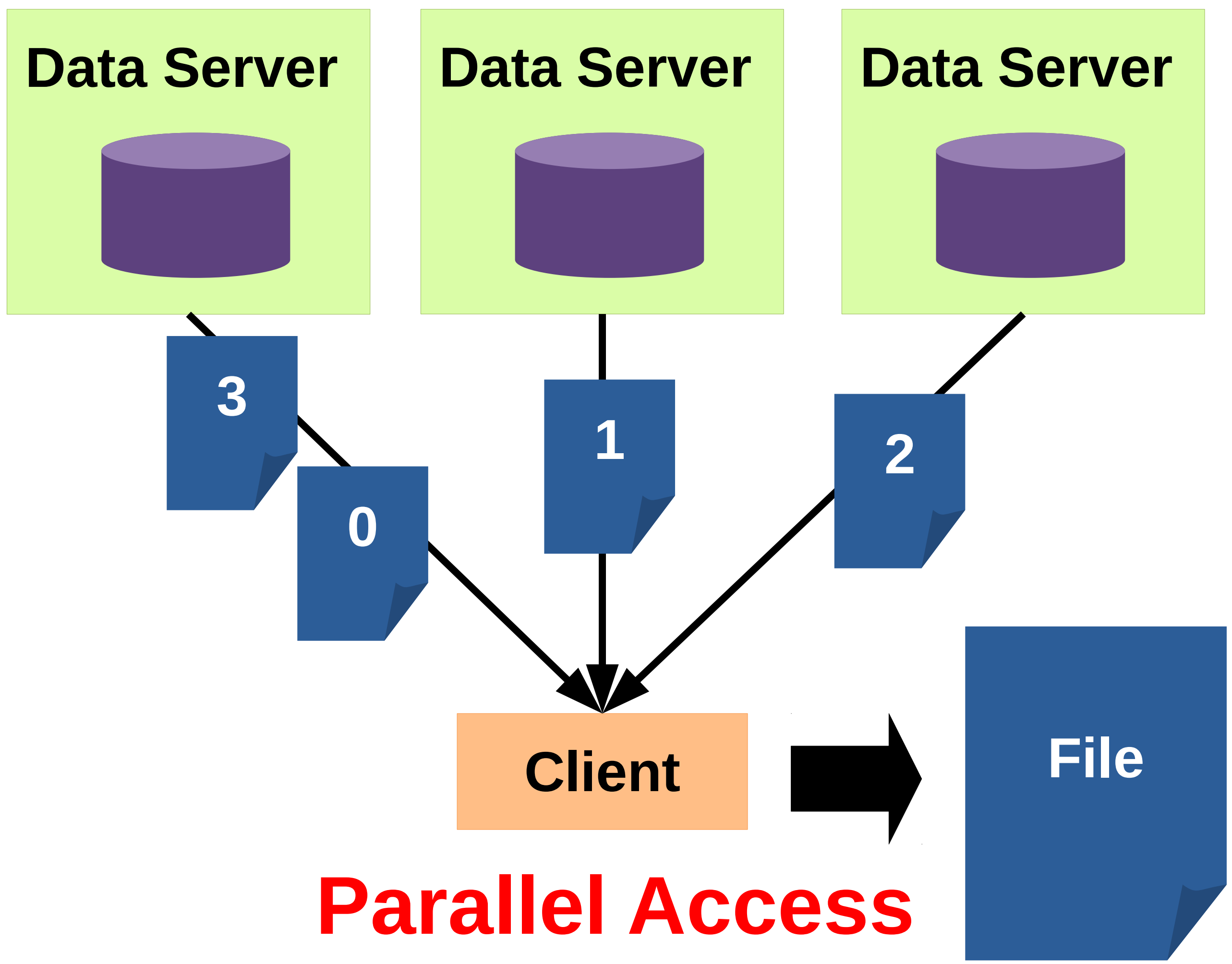
Parallel Access

Parallel Access



Parallel Access

Parallel Access



Performance Issues

- Some **access patterns** are known to present **poor performance**:
  - Small and sparse accesses
  - Small files, large number of files
  - “Out-or-order” accesses (by offset order)
  - Accesses not aligned with the stripe size
  - Concurrency on the access
  - ...

I/O Optimizations

- Several techniques try to **adapt the applications' access patterns** to improve performance.
  - Collective I/O
  - Requests reordering and aggregation
  - I/O forwarding and offloading
  - ...

Example: I/O Scheduling

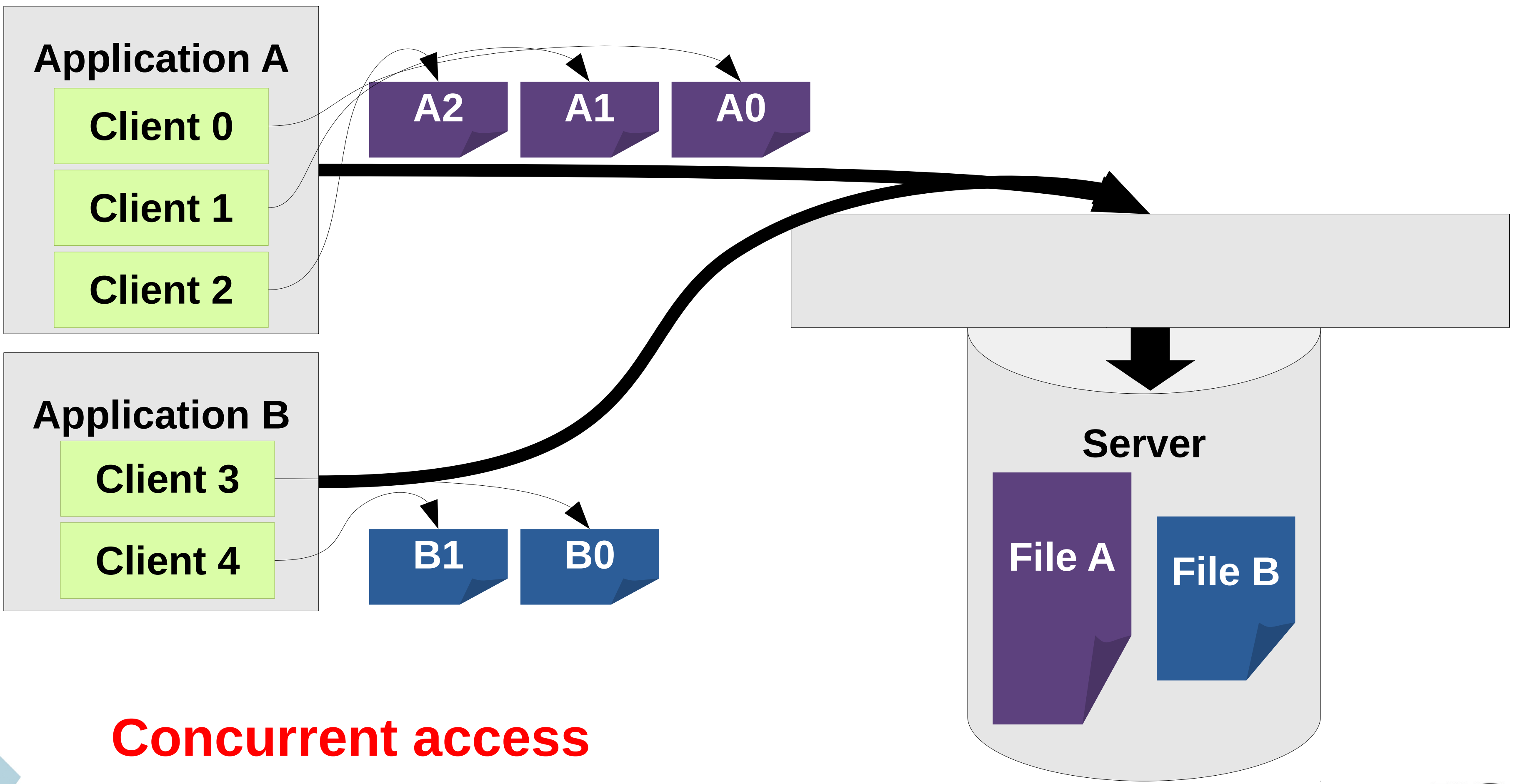
- Applications **access concurrently** the shared file system infrastructure

- **I/O Scheduling**: schedule requests to the file system in order to **minimize interference**

LIG

.inf INSTITUTO DE INFORMÁTICA UFRGS

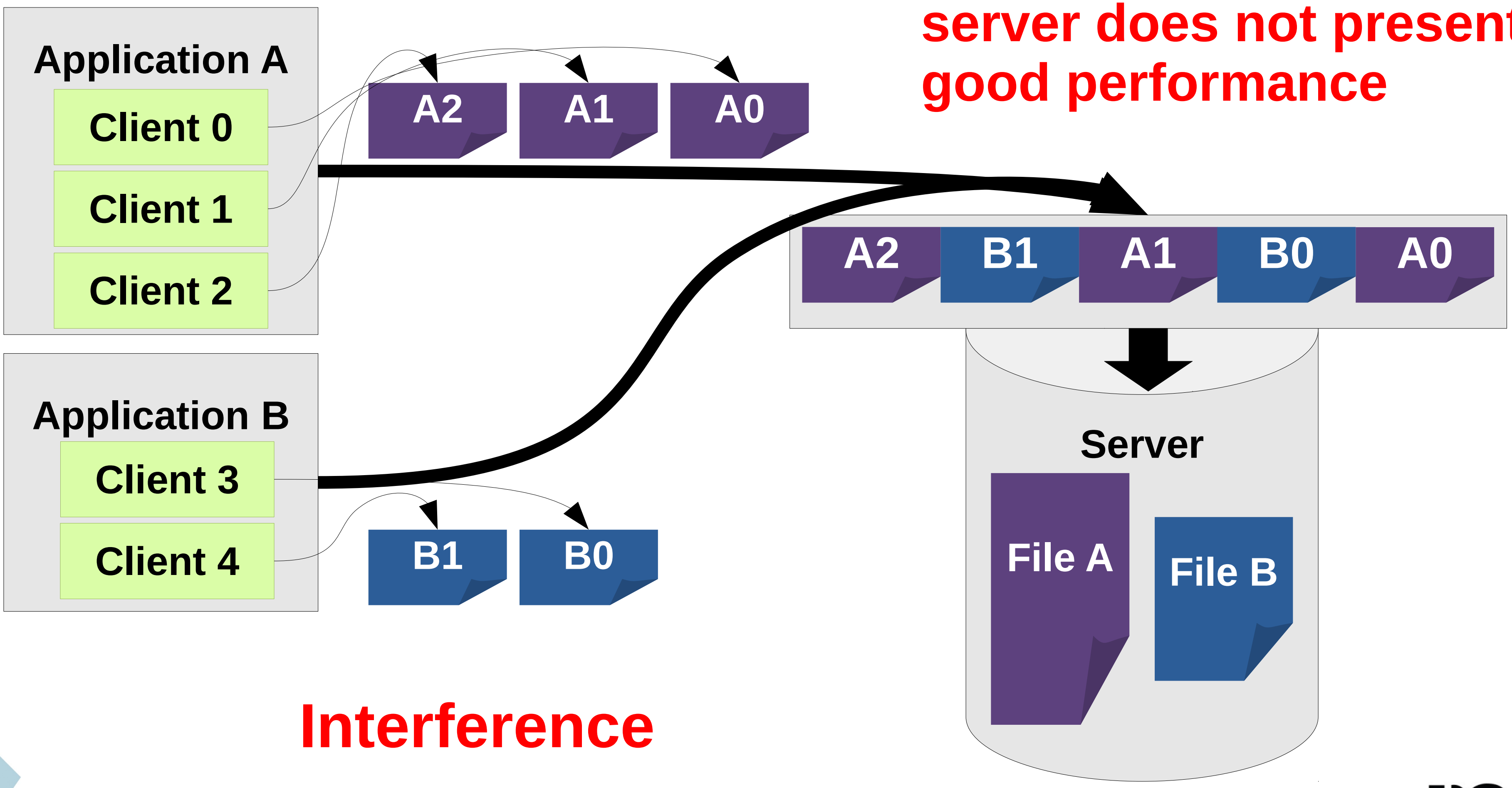
I/O Scheduling



Concurrent access

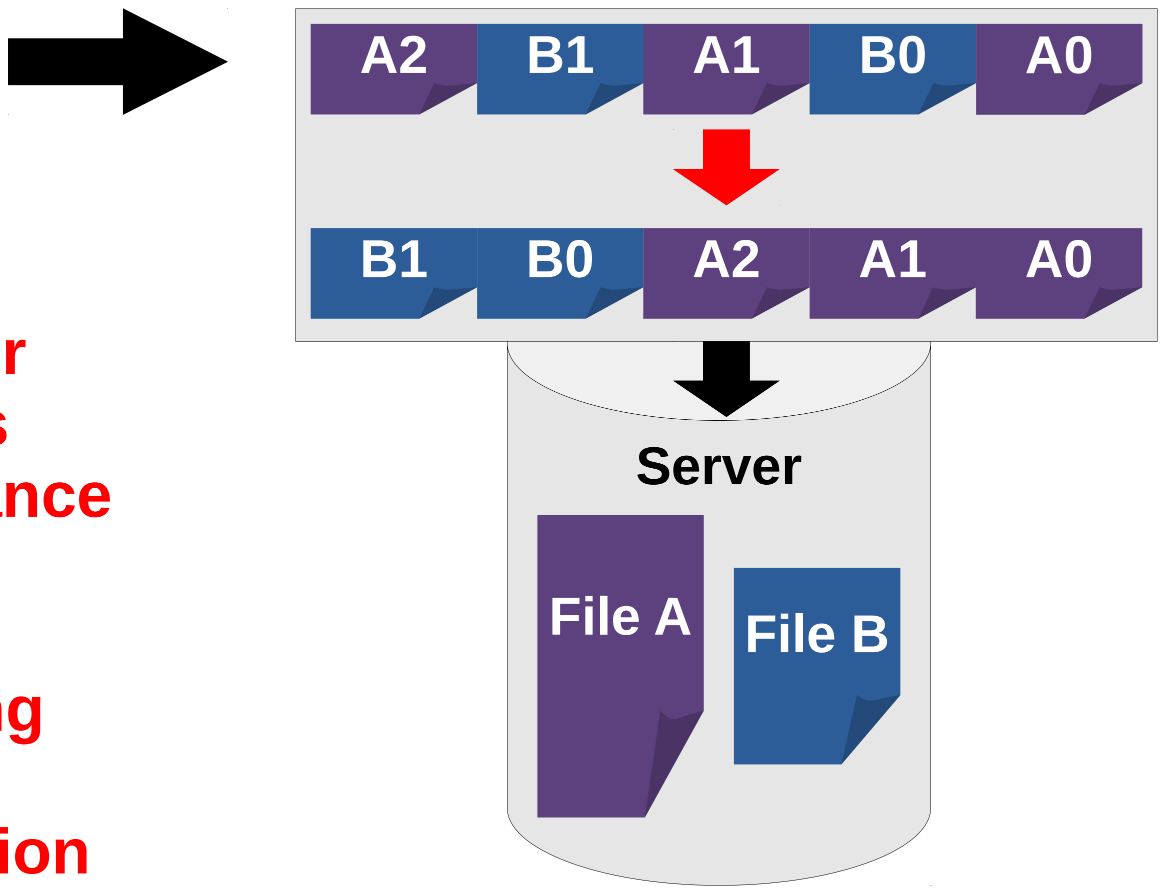
I/O Scheduling

Access pattern at the server does not present good performance



Interference

I/O Scheduling



The scheduler improves performance through requests reordering and aggregation

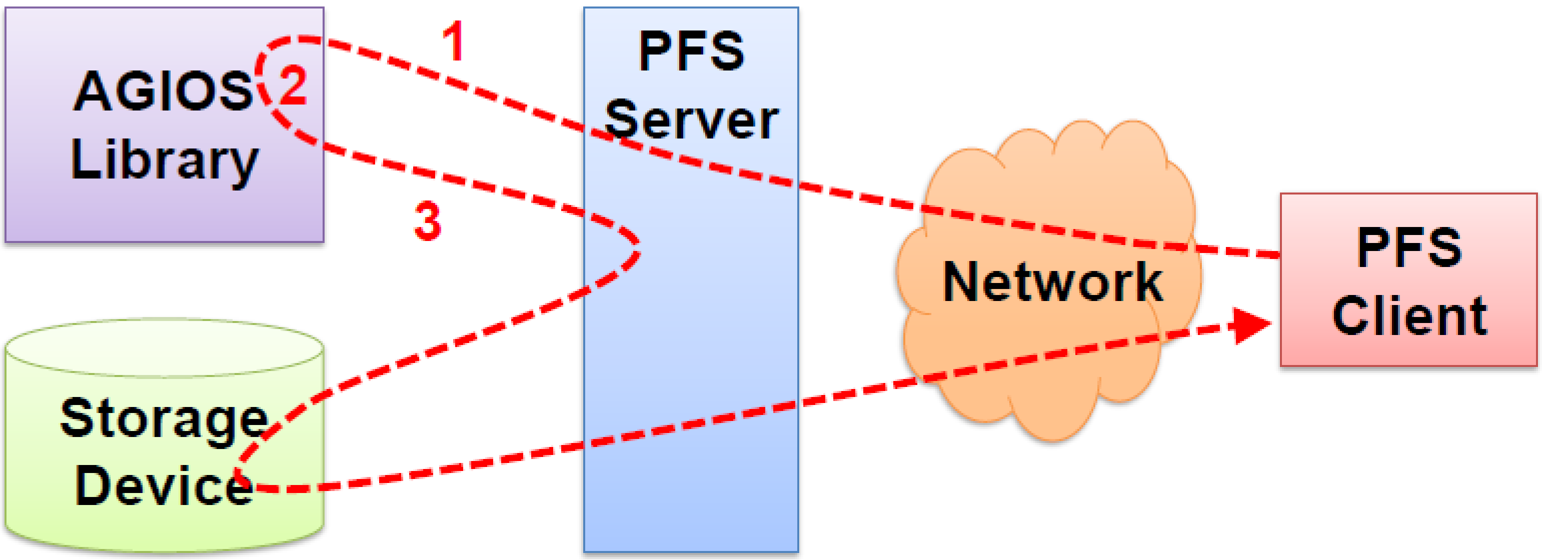
I/O Research @ GPPD UFRGS

INSTITUTO DE INFORMÁTICA UFRGS

AGIOS

- **I/O Scheduling Library** developed in our research group
- Easily included in parallel file systems
- **aIOLi algorithm** for scheduling

AGIOS



1. agios_add_request()
2. scheduling algorithm
3. “process request(s)” callback

Application-aware

- The file system servers do not have information about the application
  - **Information is lost on the I/O stack**

- Optimizations could be **smarter** with information about access patterns

AGIOS

- AGIOS = **Application-guided I/O Scheduler**
- Include information about the application on the scheduler
  - Through **traces** from previous executions
- Use information to guide the scheduler's choices
  - **Wait for incoming requests** in order to **aggregate more**

[Boito et al. 2013]

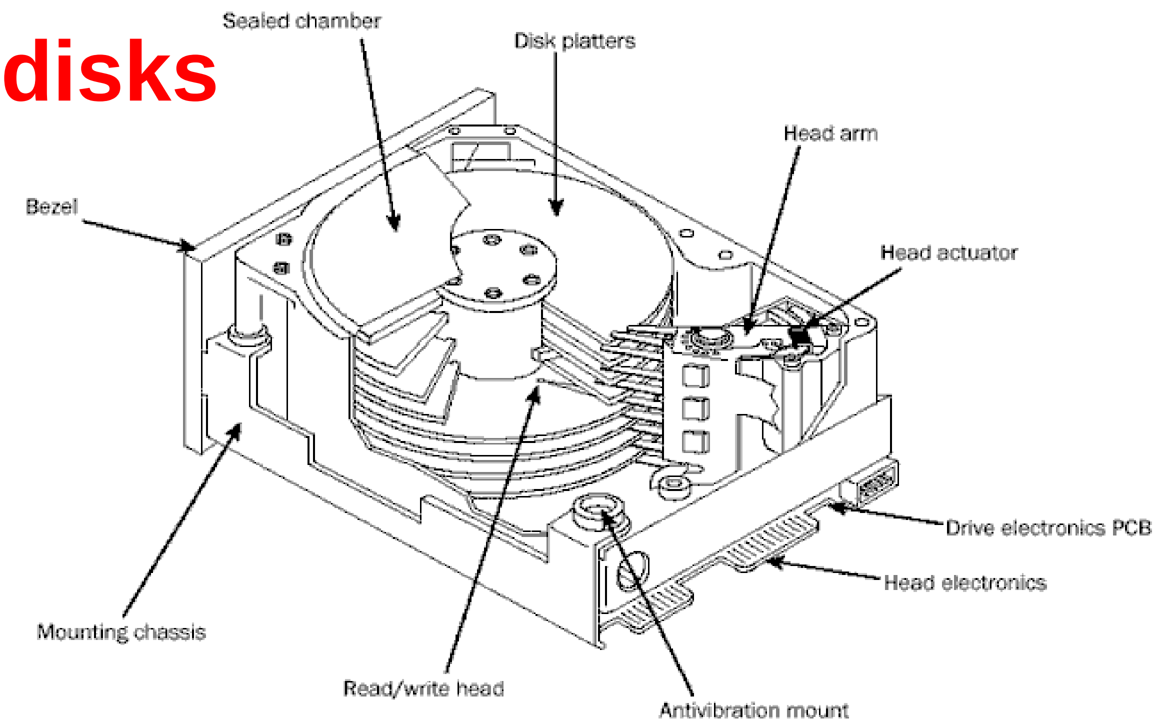
AGIOS – Performance Improvements

- Aggregations ~21% bigger
- **Performance improvements of ~25%** on average
- **Over the base** scheduling algorithm (aIOLi)
  - 46.3% on average over not using the scheduler

[Boito et al. 2013]

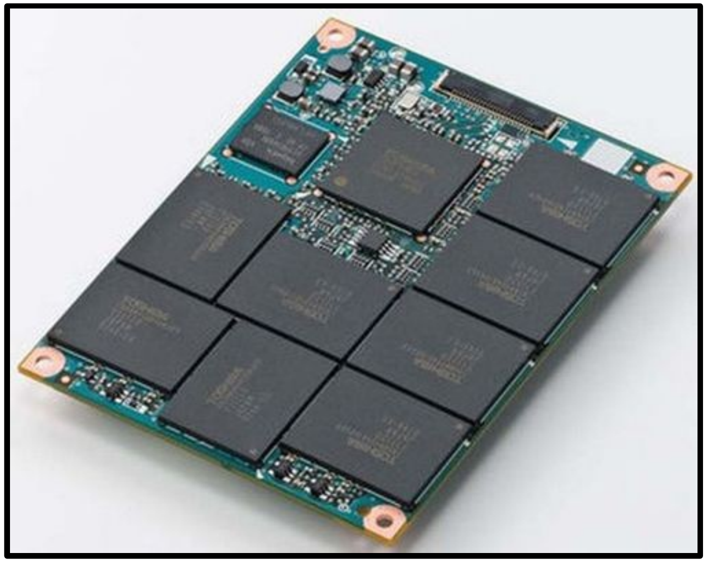
I/O Optimizations

- Several optimizations, like the scheduler, work on the **assumption that contiguous accesses are better than non-contiguous**.
- Developed for **hard disks**
  - **Seek** costs



Solid State Drive (SSD)

- Non-volatile flash-based (mostly) storage
- **No moving components**
  - more resistance to physical shocks
  - less noise
  - less heat dissipation
  - less energy consumption
- **Difference between sequential and random accesses becomes less important**



Sequential to Random Ratio

	Read Ratio	Write Ratio
HDD	143.7	66.8
SSD_1	11.0	3.1
SSD_2	9.2	328.0
SSD_3	2.4	151.6
SSD_4	1.1	1.3
SSD_5	3.2	1.5

Table from [Rajimwale, Prabhakaran and Davis 2009]

- The sequential to random access time ratio is not always smaller on SSDs than on HDDs
- **We cannot easily make assumptions about their performance**

Ongoing Research @ GPPD

- **Storage devices profiling**

- Use information about the devices in order to **select optimizations** that will improve performance

Ongoing Research @ GPPD

- **Hybrid storage infrastructures**
  - Larger, slower devices for storage space
  - Smaller, faster devices for access speed

- Management is done by the file system

Final Remarks

INSTITUTO
DE INFORMÁTICA
UFRGS

Final Remarks

- **I/O is an important issue on the path to exascale**
  - Applications have their performance impaired by I/O operations
- **Performance** depends on the access pattern
  - Several optimizations try to **adjust the access pattern**

Final Remarks

- With new technologies, assumptions on devices' performance cannot be easily made

- **All layers benefit from more information in order to make better decisions**

LIG

.inf INSTITUTO DE INFORMÁTICA UFRGS

I/O Research @ GPPD UFRGS

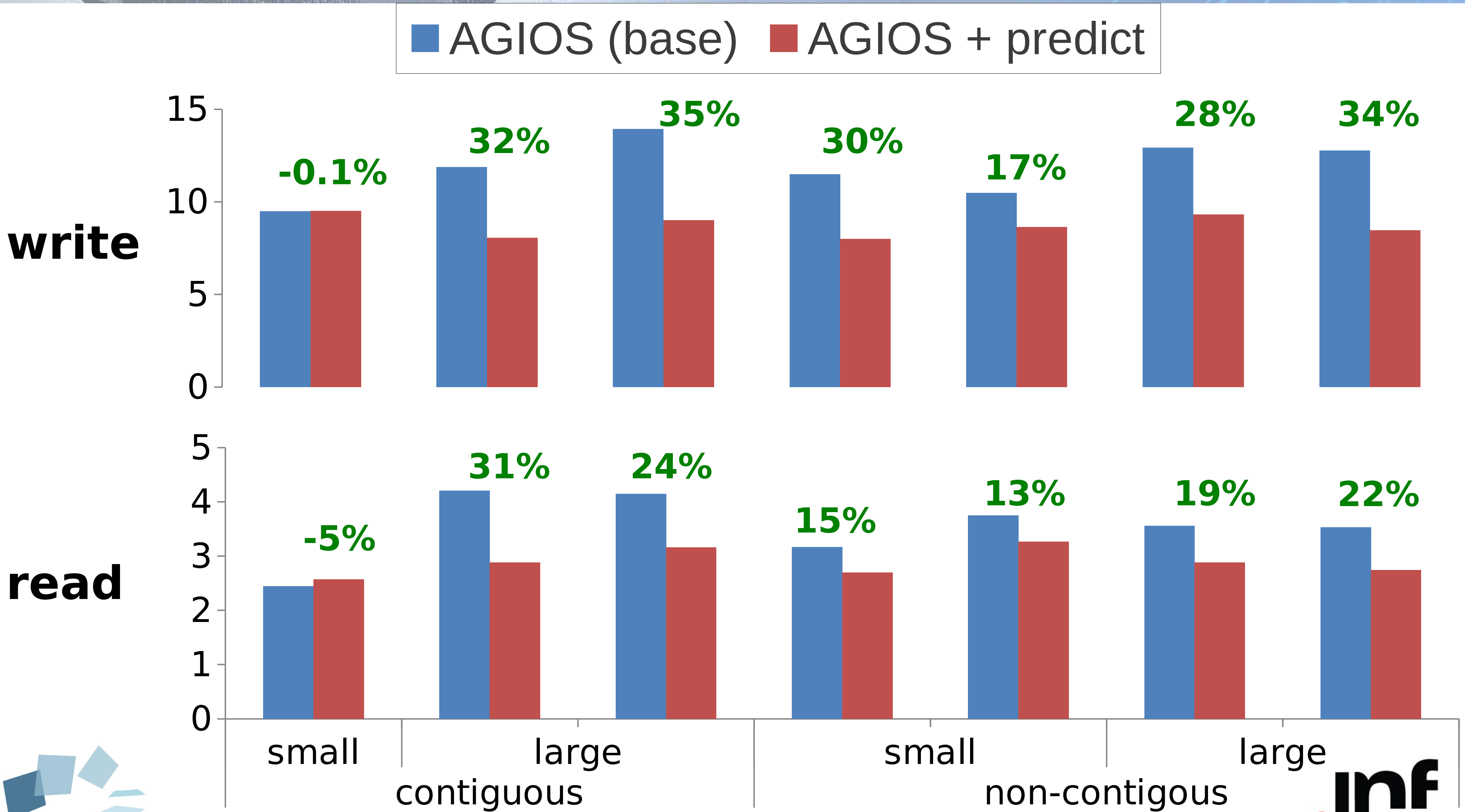
Francieli Zanon Boito
Rodrigo Virote Kassick

Philippe O. A. Navaux
Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil

Yves Denneulin
INRIA – LIG - University of Grenoble, France

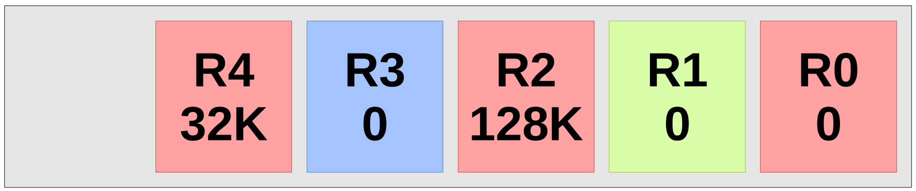
INSTITUTO DE INFORMÁTICA UFRGS

AGIOS – Performance Improvements



[Boito et al. 2013]

aIOLi scheduling algorithm



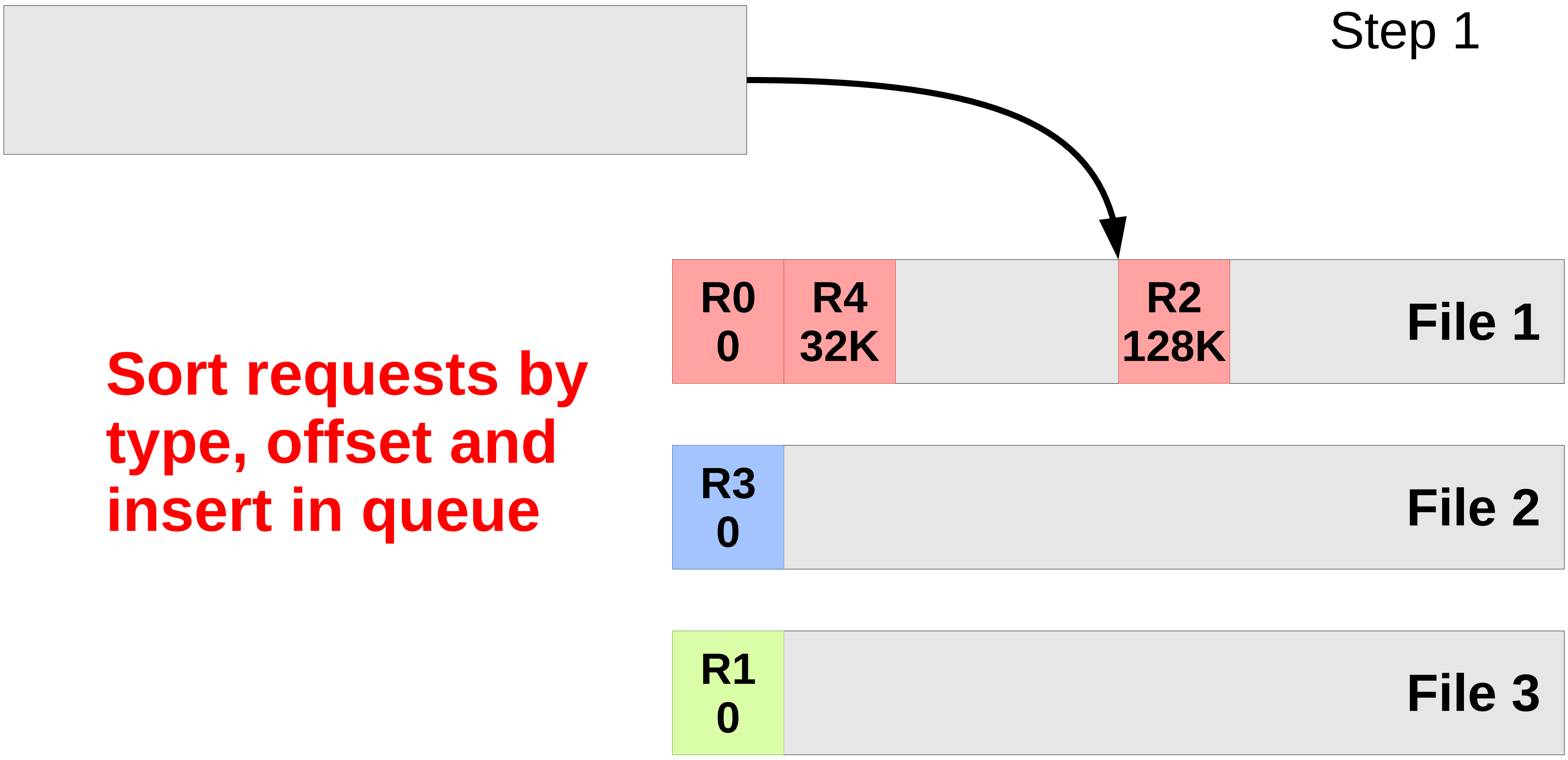
Requests of 32KB offset

Step 1

[Lebre et al. 2006]

aIOLi scheduling algorithm

Step 1

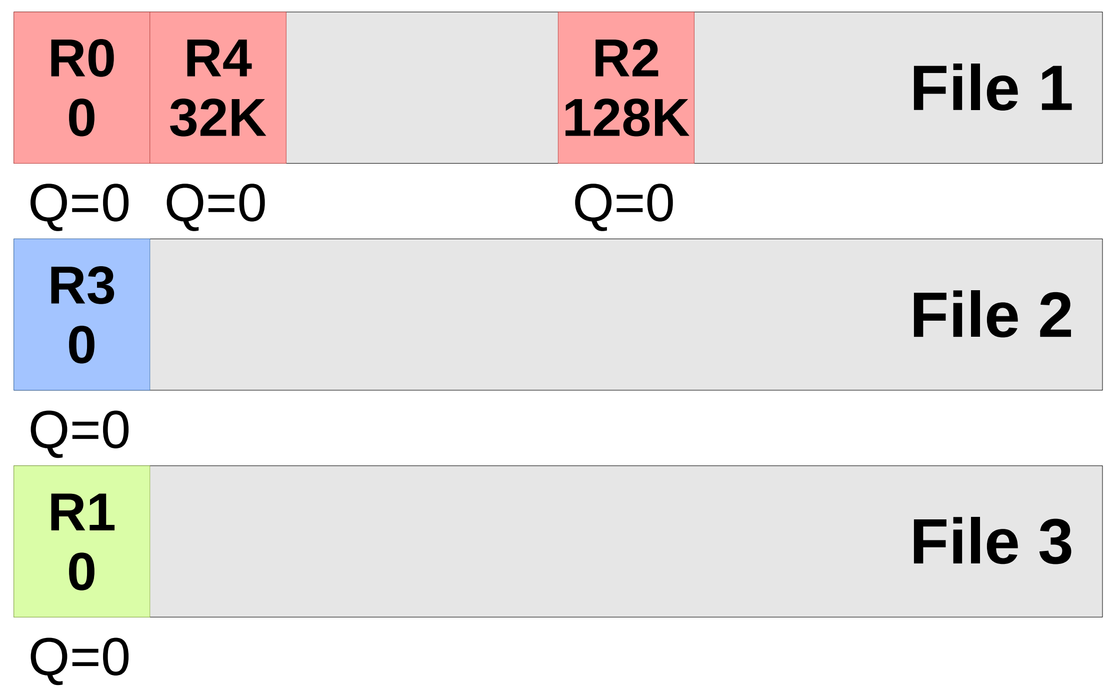


Sort requests by type, offset and insert in queue

aIOLi scheduling algorithm

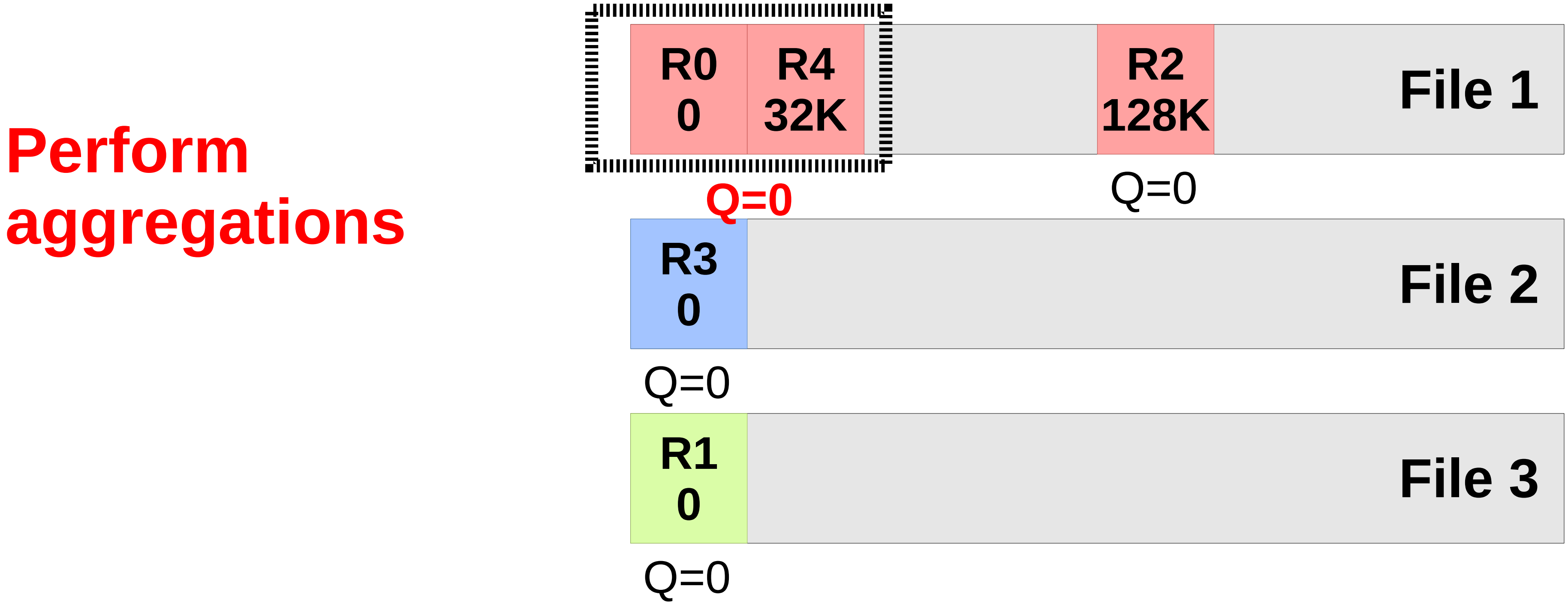
Step 1

Quantum = 0



aIOLi scheduling algorithm

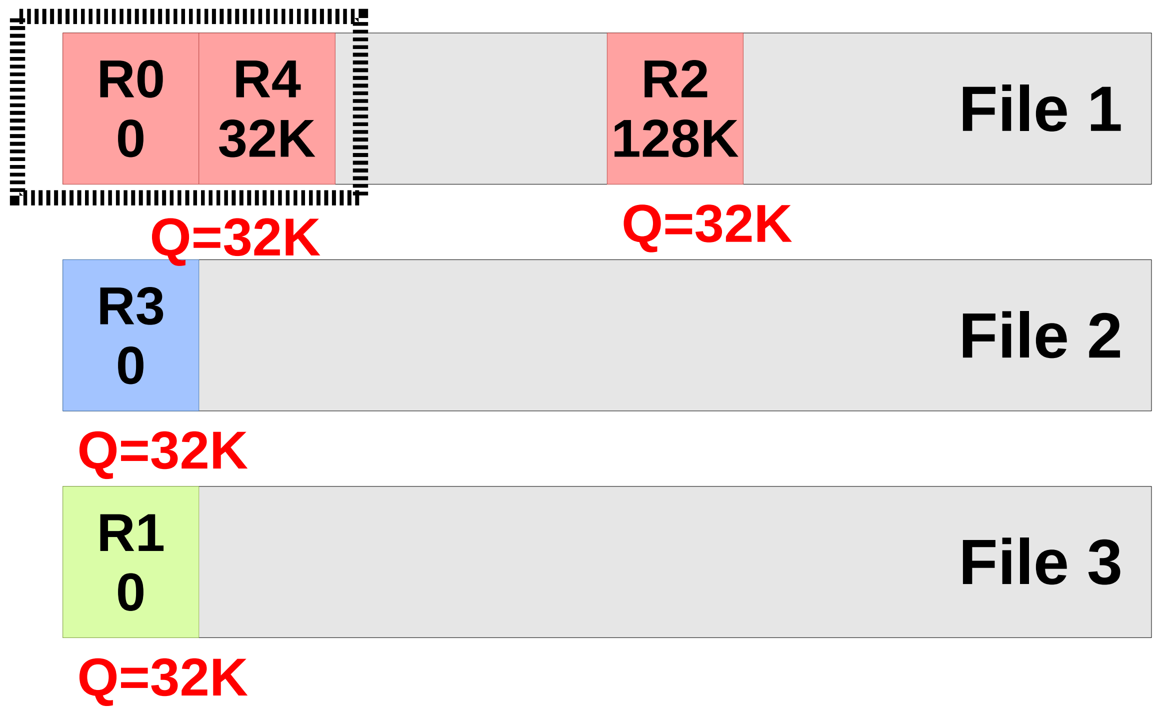
Step 1



aIOLi scheduling algorithm

Step 1

Quanta are increased by a fixed value

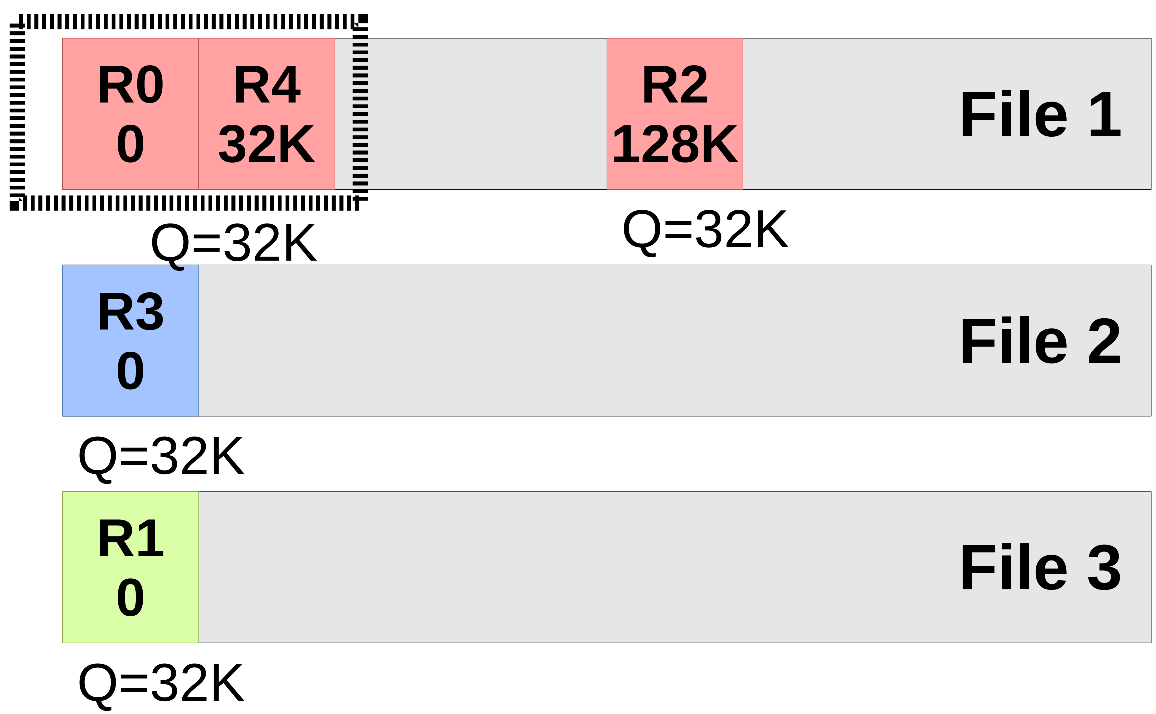


aIOLi scheduling algorithm

Step 1

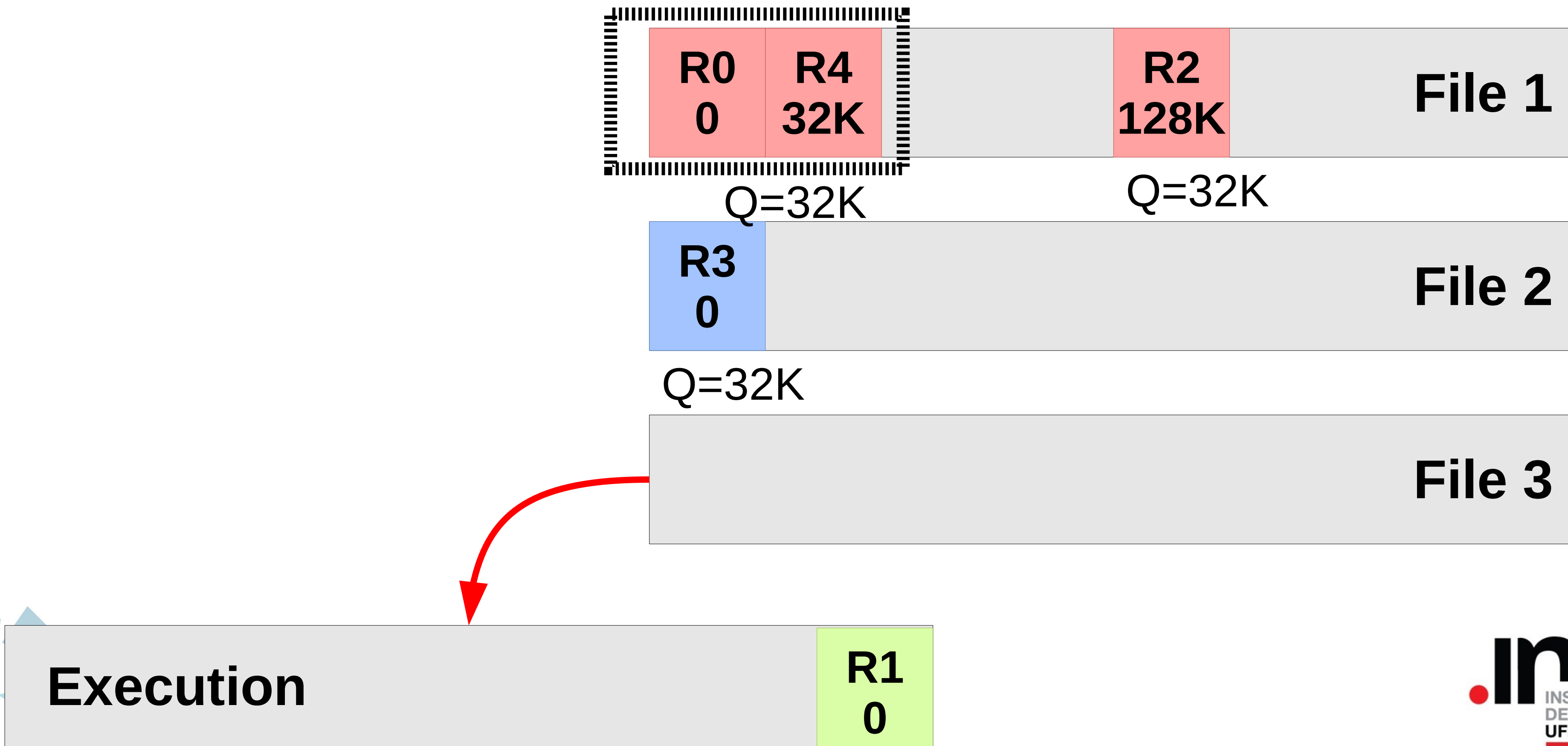
Select request

- by offset order
- FIFO between queues
- quantum is large enough for the request size

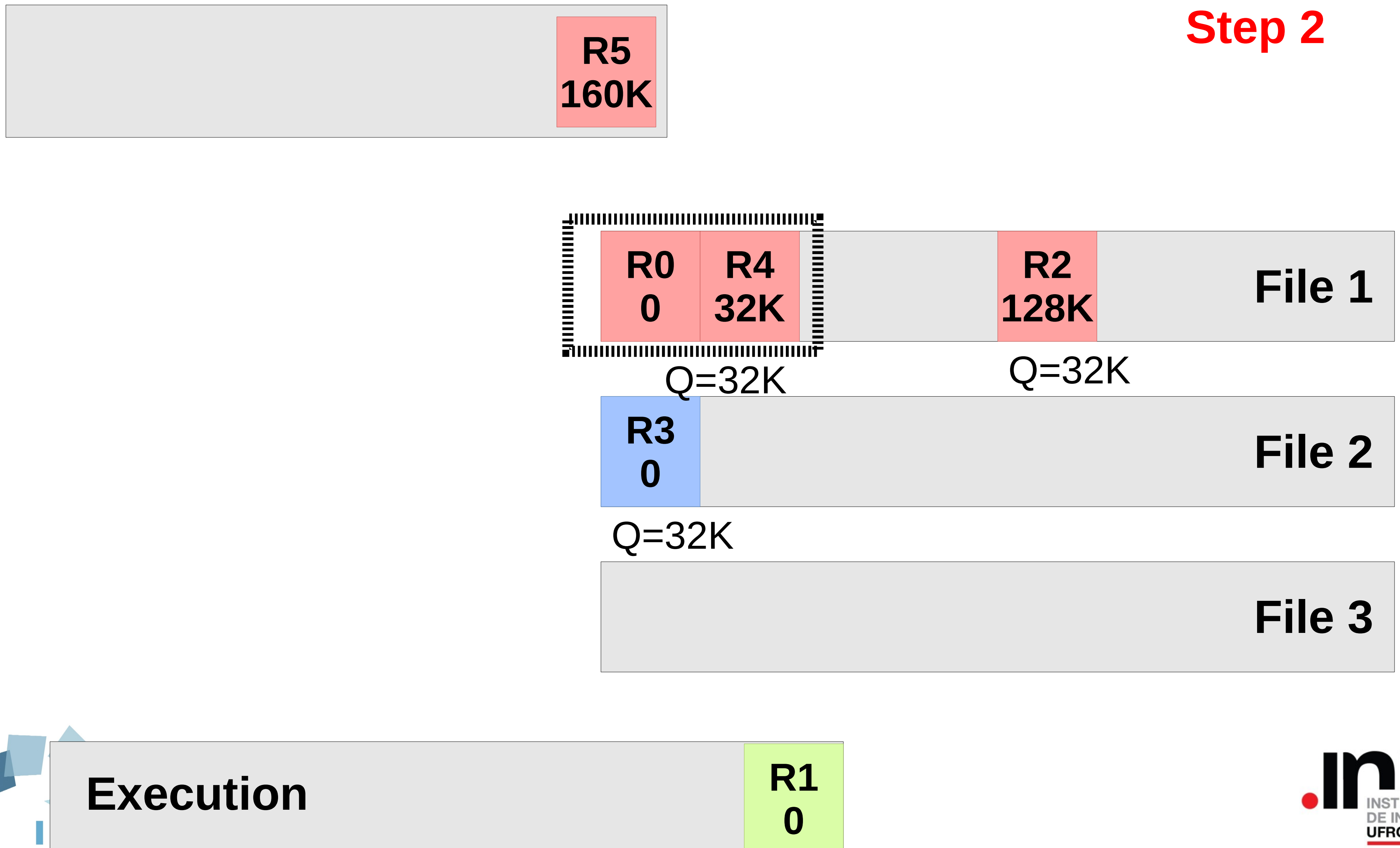


aIOLi scheduling algorithm

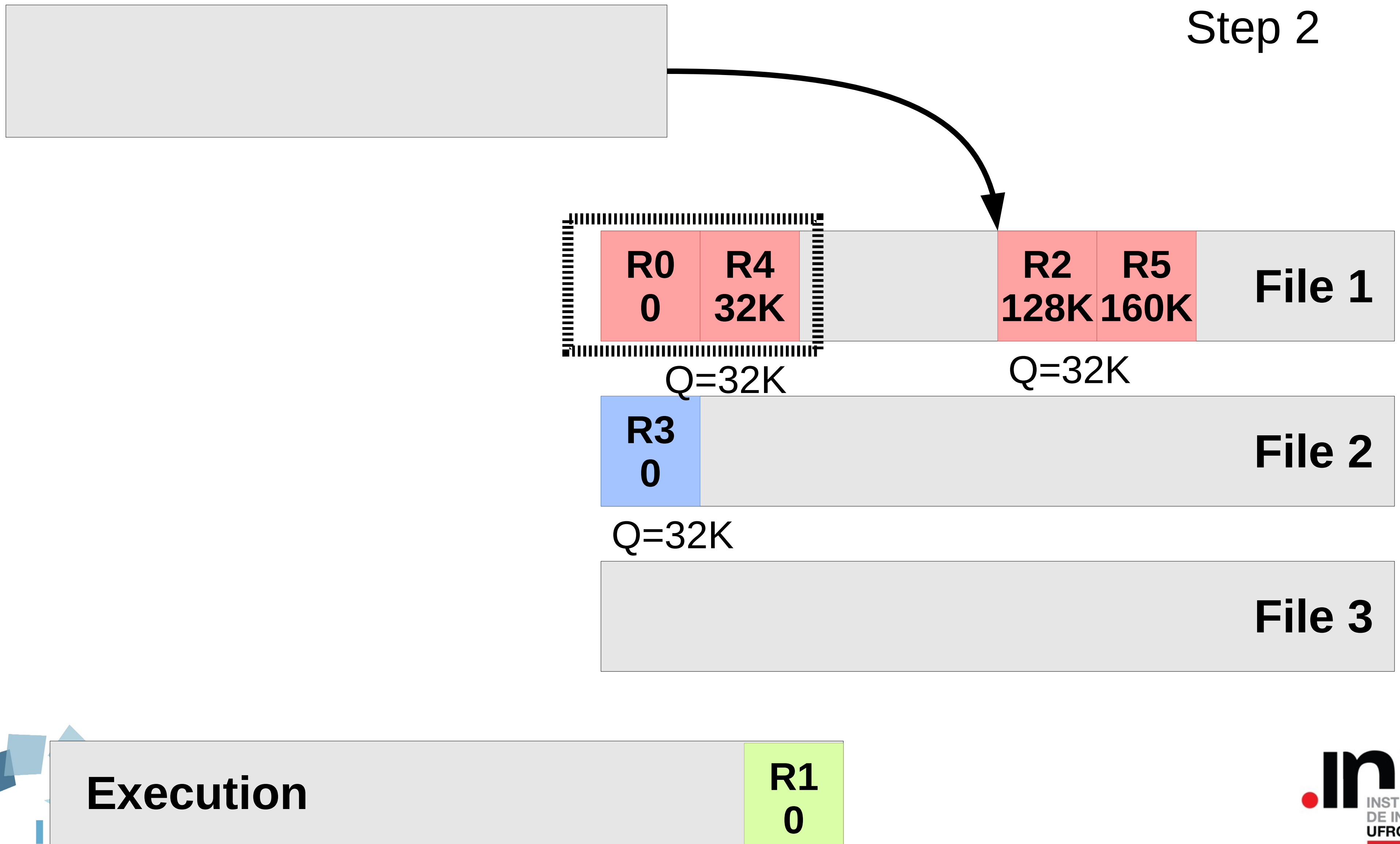
Step 1



aIOLi scheduling algorithm

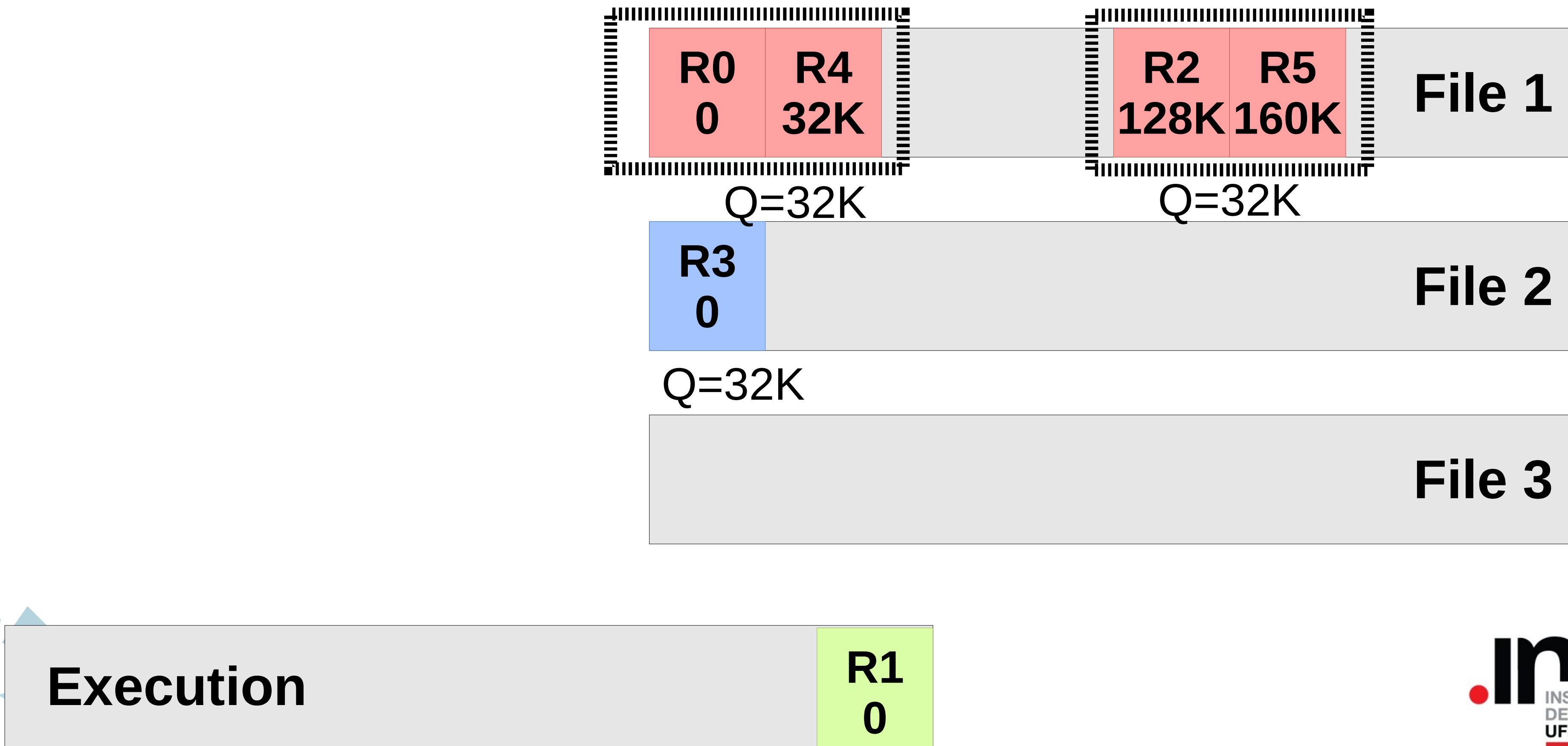


aIOLi scheduling algorithm



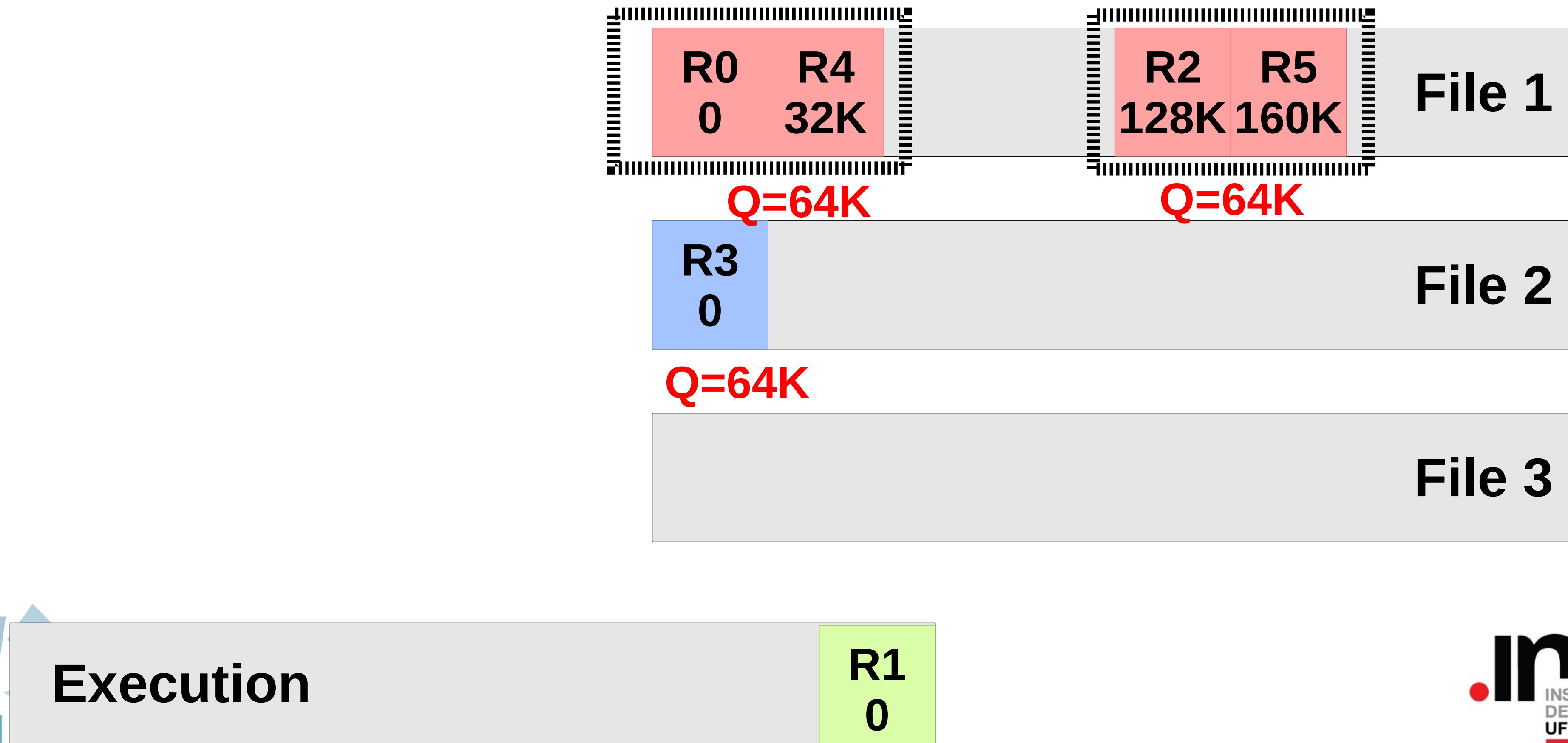
aIOLi scheduling algorithm

Step 2

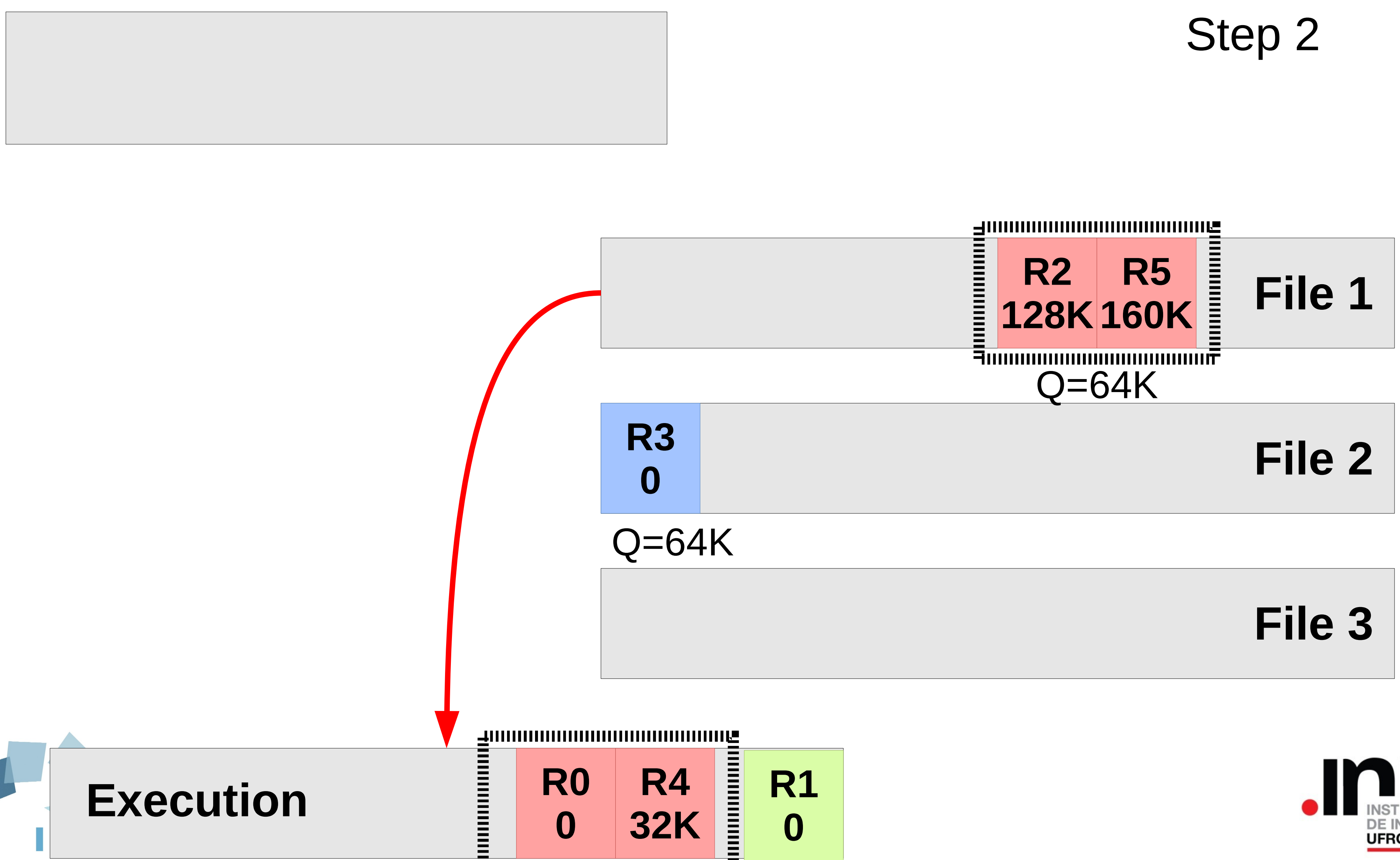


aIOLi scheduling algorithm

Step 2



aIOLi scheduling algorithm



aIOLi scheduling algorithm

File 1

...

File 2

File 3

Execution